UNIVERSITÄT
HEIDELBERG
ZUKUNFT
SEIT 1386

HOCHSCHULE HEILBRONN
TECHNIK WIRTSCHAFT INFORMATIK

Studiengang Medizinische Informatik

Analyse von Multipletts in der NMR-Spektroskopie

Abschlussarbeit zur Erlangung des akademischen Grades
Diplom-Informatikerin der Medizin
(Dipl.-Inf. Med.)

vorgelegt von
Christiane Schöpflin

November 2016

Referent:	Prof. Dr. Oliver Kalthoff
Korreferent:	Prof. Dr. Konstantin Karaghiosoff

Danksagung

Ich möchte mich bei Herrn Prof. Dr. Oliver Kalthoff für die fachliche und inhaltliche Betreuung dieser Diplomarbeit herzlich bedanken.
Vielen Dank für die hilfreichen Impulse und unterstützenden Anregungen für meine Arbeit.

Ein weiteres Dankeschön möchte ich an Herrn Prof. Dr. Konstantin Karaghiosoff für das Übernehmen des Korreferats meiner Diplomarbeit richten.

Besonderer Dank gilt meinen Eltern, die mich während meines Studiums und vor allem in der Schlussphase auf unterschiedliche Weise unterstützt haben.

Zusammenfassung

In der Kernresonanzspektroskopie, wie auch in anderen spektroskopischen Disziplinen, unterliegt die Genauigkeit und Vollständigkeit eines Spektrums den verschiedenen Einstellungen des Messgeräts sowie dem verwendeten Gerätetypen. Um das gemessene Spektrum besser analysieren zu können, wird versucht, es an die Realität anzunähern. Eine allgemeine Annahme dazu ist, dass mehrere, sich teilweise überlagernde Peaks das Spektrum bilden. Alle Peaks verlaufen dabei gemäß einer Funktion, deren Modellparameterwerte variieren.
Es gibt verschiedene Ansätze zur Bestimmung des Spektrums, zum Beispiel die Modellanpassung und die Maximum-Entropie-Methode. Eine große, gegen Unendlich strebende Anzahl an Peaks scheint die gemessenen Spektrumdaten am besten nachzubilden. Dies ist jedoch wenig naturgetreu.
In dieser Arbeit gehe ich dem Bayes'schen Ansatz zur Datenanalyse nach, um die Peakanzahl auszumachen, für die die größte Wahrscheinlichkeit in den Messdaten liegt.
Während meiner Diplomarbeit hat sich die anfängliche Annahme, dass die Peaks nach der Lorentz-Funktion geformt sind, revidiert. Letztendlich habe ich versucht, die Bestimmung der Peakanzahl mit Peaks in Form des Pseudo-Voigt-Profils durchzuführen.
Für die Berechnung der Posterior-Wahrscheinlichkeit einer Peakanzahl war es nötig, ein multiples Integral zu lösen. Dieses entstand durch die Marginalisierung einiger Störparameter.
Ein Ziel dieser Diplomarbeit war es, das multidimensionale Integral numerisch zu berechnen. Umgesetzt werden sollte dies mit dem VEGAS-Algorithmus, der das Monte-Carlo-Verfahren zur Integration verwendet. Ich habe den Algorithmus und die Anwendungen der Arbeit in Matlab implementiert.
Um die Integration mit dem VEGAS-Algorithmus zu testen, habe ich eine Beispielanwendung zur Integration unterschiedlich dimensionaler Rosenbrock-Funktionen durchgeführt.
Die Anwendung zur Bestimmung der Peakanzahl in einem Spektrum habe ich zunächst für simulierte Daten mit zwei unterschiedlichen Formeln der Lorentzfunktion umgesetzt.
Die erste Lorentzfunktion enthält zwei Modellparameter, die Lage der Amplitude und die Halbwertsbreite, und ist zu eins normiert. Bei ihr ist die Amplitude abhängig von der Halbwertsbreite.
Die zweite Formel besteht aus drei unabhängigen Modellparametern: der Amplitude, ihrer Lage und der Halbwertsbreite. Damit ist sie für die Auswertung realer Messdaten geeigneter als die vorherige normierte Lorentzfunktion.
Bei der Anwendung mit simulierten Daten mit drei Modellparametern sowie mit gemessenen Daten und dem Modell des Pseudo-Voigt-Profils konnte die Anzahl der Peaks nicht bestimmt werden.
Die Schwierigkeit der Bestimmung der Peakanzahl mit dem VEGAS-Algorithmus lag anscheinend bei der Integration über die Amplitude. Zur Klärung des Problems habe ich die Anwendung mit dem Pseudo-Voigt-Profil und den realen Messdaten über einen anderen Lösungsweg zur numerischen Integration, mit einer Likelihood-Matrix, untersucht.
Dadurch kam die Vermutung auf, dass die Diskretisierung in y-Richtung durch das Importance Sampling des VEGAS-Algorithmus nicht konform mit der Messpunktverteilung ist. Ich habe versucht über eine Präzisionsanpassung der Amplitudenstützwerte das Problem zu lösen, was teilweise gelang.
Die in dieser Arbeit erstellte Anwendung kann zur Plausibilitätsprüfung von Ergebnissen anderer Bayes' basierter Verfahren zur Peakanzahlbestimmung dienen.
Mit geschätzten Werten für die Modellparameter aller Peaks wird die multiple Integration mit dem VEGAS-Algorithmus nicht gebraucht. Die Posterior-Wahrscheinlichkeiten können somit berechnet werden und eine quantitative Bewertung der Ergebnisse unterschiedlicher Peakzahlen für das gemessene Spektrum liefern.

Inhaltsverzeichnis

1. Einleitung

In unterschiedlichen naturwissenschaftlichen Disziplinen werden Messsignale anhand von Spektren ausgewertet. Ein Spektrum setzt sich dabei aus mehreren Peaks zusammen, die sich teilweise überlagern. Sie formen sogenannte Multipletts.
Durch die quantitative Analyse können vielfältige Rückschlüsse über das untersuchte Objekt gezogen werden. Mit Hilfe der Kernspinresonanzspektroskopie lassen sich zum Beispiel Molekülkonzentrationen in einer Stoffprobe ermitteln. Für Messungen mit Magnetresonanz gibt es verschiedene Gerätearten, die in der Art der verwendeten Magnete, ihrer Größe und ihrem Preis variieren.
Die teureren Messgeräte enthalten einen supraleitenden Elektromagneten, der das Hauptmagnetfeld erzeugt. Hinsichtlich der Superleitfähigkeit sollte der Strom, der das elektrische Magnetfeld produziert, ohne Widerstand und damit verlustfrei und konstant fließen. Um das zu gewährleisten, muss der Elektromagnet auf 4,2 Kelvin ($=$ -269 $°C$) gekühlt werden [1]. Diese Temperatur wird mithilfe von flüssigem Helium und flüssigem Stickstoff erreicht. Durch die notwendige Kühlung ist ein solcher Messapparat groß und hochpreisig.
Eine mobilere und kostengünstige Alternative dazu bieten sogenannte Bench-top-Geräte.
Sie verwenden Permanentmagnete, um das Hauptmagnetfeld für die Messung zu erzeugen.
Meine Arbeit bezieht sich auf die Anwendung der Kernresonanzspektroskopie zur Signalanalyse.
Genaueres zu dieser Methode der Messung und Auswertung beschreibe ich in Abschnitt 2.1.1.

1.1. Motivation

Ein Kernresonanzspektrometer kann entsprechend der enthaltenden Magnetart unterschiedlich große Feldstärken für das homogene Hauptmagnetfeld erreichen. Elektromagnete bilden mit ihrer hohen Feldstärke Magnetfelder um die 20 Tesla (T)[1][1].
Verglichen dazu ist die Feldstärke von Permanentmagneten gering. In mobilen NMR-Spektrometern werden Magnete aus $Nd_2Fe_{14}B$ verwendet, die eine magnetische Flussdichte von 2 T [3] erzielen.
Je höher die magnetische Feldstärke bzw. Flussdichte ist, umso eindeutiger erkennbar ist die Aufspaltung der Multipletts. Durch unterschiedliche Messeinstellungen kann es zudem dazu kommen, dass Signale nicht gemessen werden oder zusätzliche Signale in der Messung auftreten, wie zum Beispiel Rauschen. Messdaten eines Spektrums können daher unvollständig und unterschiedlich stark verrauscht sein.
Für die Spektralanalyse ist es aus diesen Gründen wünschenswert, die einzelnen Peaks und damit den Verlauf des wahren Spektrums zu bestimmen. Die Annäherung eines Spektrums erfolgt bisher überwiegend mittels Modellanpassung. Je mehr Peaks in das Modell eingehen, umso genauer scheint dieses das Spektrum wiederzugeben.
In einer Stoffprobe ist die Anzahl verschiedener Stoffe natürlicherweise begrenzt. Eine gegen Unendlich strebende Zahl von Peaks ist für die Datenanalyse daher nicht sinnvoll.
Um den Verlauf des wahren Spektrums wirklichkeitsgetreuer ermitteln zu können, ist es daher hilfreich die Anzahl der Peaks festzustellen.

Eine Methode zur Bestimmung der Peakanzahl in einem Spektrum beschreiben D. S. Sivia und C. J. Carlile ihrem Artikel “Molecular spectroscopy and Bayesian spectral analysis - how many lines are there?” [4]. Sie verwenden dazu Ansätze der Bayes'schen Datenanalyse und der Wahrscheinlichkeitstheorie. In Abschnitt 2.3 führe ich ihre Herangehensweise genauer aus.

[1] Die magnetische Feldstärke (H) ist proportional zur magnetischen Flussdichte B $= \mu_0 \cdot \mu_r \cdot H$, wobei μ_0 eine universelle Naturkonstante und μ_r die Materialkonstante darstellt [2].

1.2. Zielsetzung

In meiner Arbeit gehe ich dem gleichen Ansatz wie Sivia und Carlile nach. Ein Grundgedanke ist, die Überlegungen und Annahmen quantitativ mit den gemessenen Daten auszuwerten und dadurch die beste Schätzung zu erzielen.

Einige Annahmen für meine Anwendung sind abweichend gegenüber denen von Sivia und Carlile. Sie gehen davon aus, dass die Peaks eines Spektrums gaußförmig sind. Die Vermutung, die ich umsetzen möchte, ist ein Peakverlauf gemäß der Lorentz-Funktion.
Bezüglich der spezifischen Breite[2] der Peaks nehmen Sivia und Carlile an, dass sie für alle Peaks eines Spektrums gleich ist. Des Weiteren verwenden sie dafür feste Werte, die sie zuvor schätzen. Bei der Methode dieser Diplomarbeit ist die Breite der Peaks variabel. Die einzelnen Peaks können unterschiedliche Halbwertsbreiten annehmen.

Die Peakanzahl ermittele ich, wie Sivia und Carlile, mit Hilfe der Wahrscheinlichkeitstheorie. Die Überlegungen und Rechenschritte dazu werden in den Abschnitten 2.3 und 3.1 erklärt.
Ein Teil der Berechnungen bildet ein multiples Integral.
Sivia und Carlile lösen dieses näherungsweise über eine Taylorentwicklung.
Ich möchte das multidimensionale Integral numerisch bestimmen. Verschiedene Vorgehensweisen zur Berechnung eines multiplen Integrals und meine Umsetzung beschreibe ich in Abschnitt 3.2.

Meine Arbeit soll eine Hilfestellung zur Annäherung an das wahre Spektrum bieten. Durch die Feststellung der endlichen Peakanzahl, die für das gemessene Spektrum am wahrscheinlichsten ist, möchte ich dies realisieren.

[2] Mit spezifischer Breite ist die Halbwertsbreite gemeint. Diese wird in der Gaußfunktion durch σ und in der Lorentz-Funktion durch Γ dargestellt.

2. Grundlagen

Zunächst möchte ich das wissenschaftliche Umfeld der Kernspinresonanz-Spektroskopie, für die ich die Anwendung meiner Arbeit entwickle, beschreiben. Ich erläutere woher die Messung stammt und wie das Spektrum, das analysiert werden soll, entsteht.
Für die Auswertung des Spektrums verwende ich Prinzipien der Wahrscheinlichkeitstheorie, die ich in Abschnitt 2.2 vorstelle.
Ein wichtiger Bezugspunkt meiner Arbeit ist ein Verfahren von Sivia und Carlile zur Bestimmung der Peakanzahl in einem Spektrum. Dieses schildere ich in Abschnitt 2.3.
Ein wesentlicher Teil der Wahrscheinlichkeitsberechnung, die ich ebenso wie Sivia und Carlile durchführe, bildet ein multidimensionales Integral. In Abschnitt 2.4 erkläre ich das Monte-Carlo-Verfahren, mit dem ich das Integral in meiner Arbeit löse. Für die Realisierung des Verfahrens setze ich den VEGAS-Algorithmus von G. P. Lepage ein, den ich in Abschnitt 2.5 beschreibe.

2.1. Molekülspektroskopie

Das Teilgebiet der Spektroskopie schließt unterschiedliche Analysemethoden ein, die sich mit den Wechselwirkungen zwischen Molekülen und elektromagnetischen Feldern befassen.
Je nach Methodenansatz werden Rotations-, Schwingungs- oder Elektronenzustände in den Molekülen eines chemischen Stoffes angeregt.
Die Reaktion auf diese Veränderungen in und um die Moleküle wird in Form eines Spektrums ausgewertet. Das Ziel der Untersuchung ist es, die molekularen Eigenschaften des Stoffes zu charakterisieren und seine atomaren Bestandteile zu identifizieren.

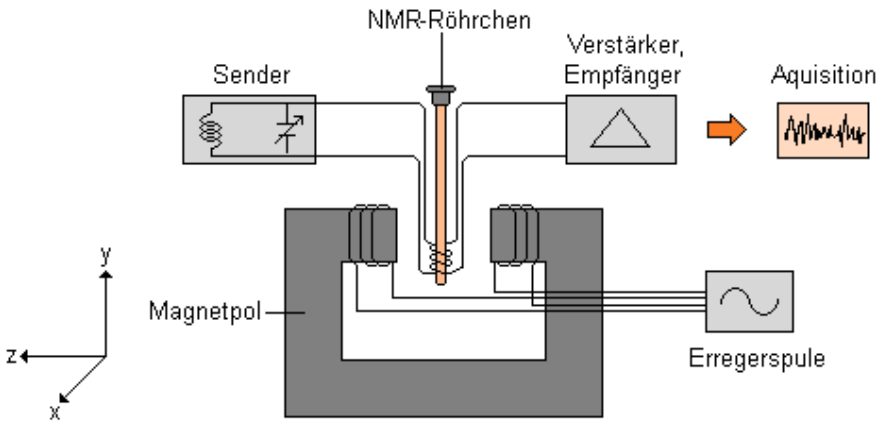
2.1.1. Kernspinresonanz-Spektroskopie

Die Kernspinresonanz-Spektroskopie, auch NMR[1]-Spektroskopie genannt, ist eine Methode der Molekülspektroskopie. Mit ihrer Hilfe wird die elektrische Umgebung einzelner Atome und Wechselwirkungen zu ihren Nachbaratomen untersucht.
Die Messungen dienen der Analyse der chemischen Struktur und der Dynamik von Molekülen eines Stoffes. Des Weiteren wird das Verfahren zur Bestimmung von Molekülkonzentrationen in einem Stoff verwendet. [5]

Die Messung

Um die Messung durchführen zu können, werden flüssige Proben benötigt. Der zu untersuchende Stoff wird dafür gegebenenfalls mit geeigneten Lösungsmitteln präpariert.
Für den Messvorgang wird die Stoffprobe in ein homogenes magnetisches Feld gebracht.
Teilchen und Atomkerne, die einen Spin ungleich null besitzen, verhalten sich in einem Magnetfeld als hätten sie ein magnetisches Moment. Nur solche Atomkerne können mit NMR untersucht werden. Beispiele dafür sind: ^{1}H, ^{11}B, ^{13}C, ^{15}N, ^{17}O, ^{19}F, ^{31}P [6].
Die Spins der Atome in der Probe richten sich in Richtung des einwirkenden Hauptmagnetfeldes aus. Ein weiteres elektromagnetisches Feld ist Teil des Messaufbaus. Es wird mit einer Induktionsspule erzeugt, die um die Probe platziert wird. Auf diese Weise entsteht ein zum Hauptmagnetfeld senkrecht orientiertes, magnetisches Wechselfeld.
Das Wechselfeld bewirkt, dass sich die Spinorientierungen der Probe, die mit der einstrahlenden

[1] Abkürzung von engl.: nuclear magnetic resonance



Quelle: Teach/Me Instrumentelle Analytik [7]

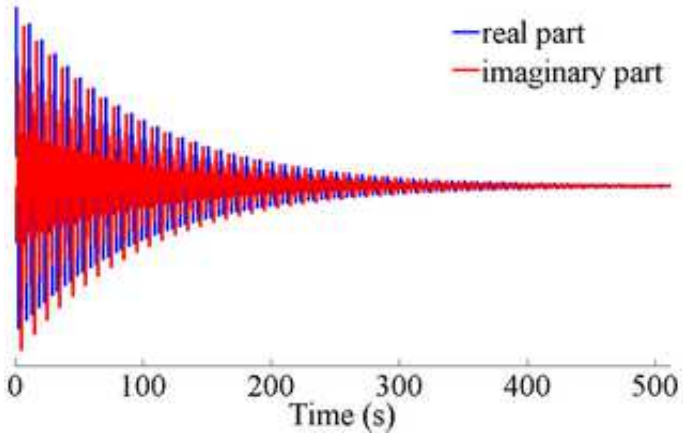
Abbildung 2.1.: Schematischer Aufbau eines NMR-Spektrometers

Energie in Resonanz stehen, verändern. Je länger der Impuls und je höher die Leistung der Senderspule, umso weiter ist dabei die Drehung.
Sobald der Impuls endet, beginnt die Relaxation. Die Magnetisierungen der Probe nähern sich zeitverzögert dem ursprünglichen Gleichgewichtszustand im homogenen Hauptmagnetfeld. Die Energieabsorption kann mittels eines Empfängers, zum Beispiel der sendenden Induktionsspule, in Form von Radiowellen gemessen werden.
Die Resonanz in den Atomen der Probe kann auf unterschiedliche Weise ausgelöst werden. Eine Möglichkeit ist die Stärke des Hauptmagnetfeldes zu variieren.
Eine weitere stellt die Veränderung der Frequenz des Wechselfeldes bis zum Resonanzfall dar, wobei die magnetische Feldstärke des Hauptmagnetfeldes konstant ist. [8]
In modernen Messvarianten löst die Verwendung von Radiowellen-Pulsen die kontinuierlichen Wechselfelder für die Auslösung der Resonanz ab. Ein kurzer Radiowellen-Puls regt dabei Frequenzbereiche innerhalb eines Frequenzbandes zur Änderung der Orientierung des Spins an.
Während der Relaxation oszilliert jeder Kernspin mit seiner individuellen Resonanzfrequenz für kurze Zeit senkrecht zum Hauptmagnetfeld. Diese Frequenz wird Larmorfrequenz genannt [9].
Sie ist abhängig von dem einzelnen, atomar aktiven Magnetfeld. Dies kann in seiner Stärke vom Hauptmagnetfeld abweichen, da es durch die molekular nahe elektrische Umgebung bestimmt wird.

Magnetische Wechselwirkungen zwischen benachbarten Atomkernen des gleichen Moleküls und der Nachbarmoleküle bewirken, dass Atomkerne in einer Stoffprobe unterschiedliche Magnetfelder erfahren und daher spezifische Larmorfrequenzen aufweisen.

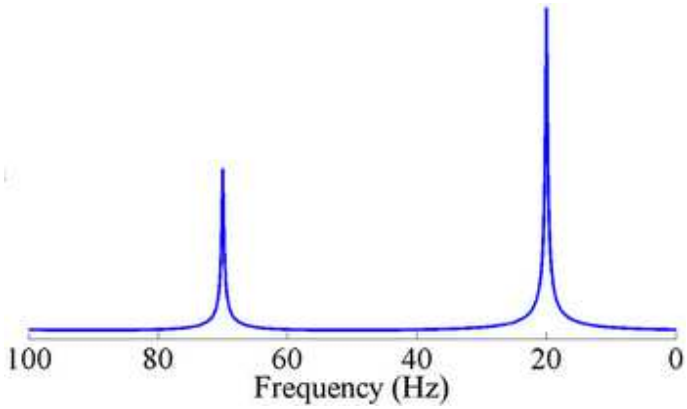
Auswertung der Messung

Gemessen wird ein Radiowellensignal in der Zeitspanne vom Ende des Impulses bis zum Erreichen des Gleichgewichtszustandes. Es zeigt den zeitlichen Abfall der Frequenzen aller zuvor angeregten Kernarten. Dieses Signal in Form einer gedämpften Schwingung wird Free Induction Decay (FID) genannt.
Durch eine Fouriertransformation kann der zeitliche Verlauf des Signals in einen frequenzabhängigen überführt werden. Dabei erhält man den realen und den imaginären Signalteil des ursprünglichen FID. Ein Beispiel für einen FID ist in Abbildung 2.2 gegeben. Der Imaginärteil des transformierten Signals wird für die Phasenkorrektur gebraucht [11]. Die andere Hälfte, der Realteil, bildet das Frequenzspektrum, zu sehen in Abbildung 2.3. Es enthält die Verteilung der individuellen Larmorfrequenzen und wird weiter analysiert.



Quelle: [10]

Abbildung 2.2.: FID



Quelle: [10]

Abbildung 2.3.: Spektrum der Resonanzfrequenzen

Da die gemessenen Resonanzfrequenzen abhängig von der gerätespezifischen Magnetfeldstärke sind, wird die chemische Verschiebung δ für die Auswertung verwendet:

$$\delta = \frac{\nu_{\text{Probe}} - \nu_{\text{Referenz}}}{\nu_{\text{Referenz}}} \cdot 10^6 \qquad (\nu := \text{Frequenz})$$

Sie gibt den relativen Abstand der Lage der Amplitude eines Peaks der Probe (Frequenz ν_{Probe}) zu der des Peaks eines beliebig gewählten Standards (Frequenz ν_{Referenz}) an. Dem Standard wird die chemische Verschiebung $\delta = 0$ zugeordnet.
Die Maßeinheit ist *ppm* (parts per million, 1 ppm $= 10^{-6}$).
Für die ^{1}H- und ^{13}C-NMR-Spektroskopie in organischen Lösungsmitteln gilt zum Beispiel Tetramethylsilan (TMS, $(CH_3)_4Si$) als innerer Standard bzw. dessen Frequenz als Referenzfrequenz. [12]
Mit Hilfe der geräteunabhängigen, relativen Größe sind Verschiebungstabellen für Kernarten erstellt worden, in denen man die entsprechende Strukturgruppe für einen δ-Wert ablesen kann.

Die Anzahl und Amplitudenlagen der Peaks sind charakteristisch für das jeweilige Molekül.
Die genaue Untersuchung des Spektrums kann daher viele Informationen über die Stoffprobe liefern.
Die gemessene Intensität der Resonanz gibt Rückschluss auf die Anzahl der Atome einer Gruppe.
Das Integral unter einem Peak gibt die Anzahl der gleichwertigen Kerne in einer Verbindung an.
Da die Atomkerne über ihre chemische Bindungen miteinander wechselwirken, kommt es zu einer Aufspaltung der Signale in Multipletts. Ihre Anzahl lässt auf die Zahl unterschiedlicher Atomgruppen schließen.
Über den Abstand zweier benachbarter Peaks eines Multipletts, der als Kopplungskonstante bezeichnet wird, kann auf die Art und den Abstand von Nachbarkernen geschlossen werden.
Mit Hilfe der Messung können Relaxationseffekte genauer betrachtet werden, die Einblick in Bezug auf vorhandene Wechselwirkungen und molekulare Bewegungen geben können.
Viele charakteristische Moleküleigenschaften können mit der NMR-Spektroskopie untersucht und zum Beispiel die genaue Zusammensetzung einer Stoffprobe bestimmt werden.

2.2. Grundlagen aus der Wahrscheinlichkeitstheorie

Die Aufgabe dieser Arbeit ist die Anzahl der Peaks in einem Spektrum genauer zu bestimmen. Hierfür wird die Wahrscheinlichkeitstheorie herangezogen, um mittels eines Modells die Wahrscheinlichkeiten unterschiedlicher Peakzahlen berechnen zu können.
In diesem Abschnitt werden die angewandten Prinzipien kurz vorgestellt.
Die Variablen a und b stellen dabei beliebige Ereignisse dar.

Produktregel

$$P(a,b) = P(a|b) \cdot P(b) \qquad (= P(b|a) \cdot P(a))$$

Die gemeinsame Wahrscheinlichkeit für das Eintreten der Ereignisse a und b ist direkt aus der Definition der bedingten Wahrscheinlichkeit ableitbar: $P(a|b) = \frac{P(a,b)}{P(b)}$ [13].

2.2.1. Bayes-Theorem

$$P(a|b) = \frac{P(b|a) \cdot P(a)}{P(b)}$$

Die Wahrscheinlichkeit für das Eintreten des Ereignisses a soll unter der Bedingung b berechnet werden. Zur Veranschaulichung des Theorems in Bezug auf die Datenanalyse kann b den Erhalt von Daten aus einer experimentellen Messung repräsentieren. Die Wahrscheinlichkeit $P(a|b)$ beschreibt einen Posterior, in diesem Beispiel die Wahrscheinlichkeit für das Ereignis a nach Datenansicht. Für

die Berechnung des Posteriors wird die Likelihood-Funktion $P(b|a)$ verwendet. Sie gibt die Wahrscheinlichkeit an, die Daten gemessen zu haben, wenn a eingetreten ist. Die Wahrscheinlichkeit des Eintretens von a unabhängig von einer vorhergehenden Datenerhebung wird in Form des Priors $P(a)$ mit der Likelihood-Funktion multipliziert. Dieses Produkt wird relativiert zum Prior der gemessenen Daten, der Daten-Evidenz $P(b)$ [14].

2.2.2. Marginalisierung

Sie wird im Umgang mit Störparametern angewendet. Wenn in eine Datenanalyse zum Beispiel die Größen X und Y eingehen, aber nur die Werte von X für die Auswertung interessant sind, kann Y mit Hilfe der Marginalisierung "weg-integriert" werden:

$$P(X) = \int_{-\infty}^{\infty} P(X,Y)\ dY$$

dabei gilt $\int_{-\infty}^{\infty} P(Y|X)\ dY = 1$ [14].

2.3. Bestehende Methode zur Berechnung der Peakanzahl eines Spektrums (von D. S. Sivia und C. J. Carlile)

Ein Ausgangspunkt für diese Arbeit ist u.a. der Artikel "Molecular spectroscopy and Bayesian spectral analysis - how many lines are there?" von Sivia und Carlile [4]. Die Ausführungen dazu sind auch als ein Beispiel in Sivias Buch "Data Analysis - A Bayesian Tutorial" in dem Kapitel zur Modellauswahl festgehalten [15].

Sie beschreiben ein von ihnen entwickeltes Verfahren, mit dem die Peakanzahl eines gemessenen Spektrums bestimmt werden kann.
Zu Beginn ihres Artikels zeigen die beiden Unterschiede des Bayes'schen Ansatzes in der Spektralanalyse gegenüber der traditionellen Modellanpassung und der Maximum-Entropie-
Methode (*MaxEnt*) auf. In der Methode der kleinsten Quadrate wird mit einer vorgegebenen Anzahl von Peaks ein Modell erstellt. Dieses wird durch Veränderung der Modellparameter (z.B. Amplituden und
deren Positionen) den Messdaten soweit angepasst bis die beste Schätzung der Parameterwerte und damit des Spektrums erreicht ist.
Bei der *MaxEnt* wird ein Teil des Wissens über das Spektrum vor der Messung verwendet, um die beste Schätzung dafür zu finden.
Die Bayes'sche Datenanalyse kombiniert das Vorwissen über ein Spektrum mit den vorhandenen Daten, die zu analysieren sind. Bei der Spektralanalyse wird zunächst die Peakanzahl gesucht, für die die größte Wahrscheinlichkeit in den Messdaten besteht. Mit dieser können dann die gesuchten Parameter des Spektrums bestimmt werden.
Dadurch erhält man die beste Schätzung des Spektrums und deren Zuverlässigkeit, gestützt durch den direkten Beweis der gemessenen Daten.
Anhand der Breite der Wahrscheinlichkeitsverteilungsfunktion um den Maximalwert kann abgelesen werden wie zuverlässig die Vorhersage ist.
Je schmaler die Spitze, umso verlässlicher ist die Schätzung.
Sivia und Carlile halten in ihrem Artikel fest, dass ein molekulares Erregungsspektrum $F(\epsilon)$ von
Natur aus positiv und additiv verteilt ist. Es stelle eine positive Verteilung dar, weil es proportional zur Anzahl molekularer Formen im Erregungszustand der Energie ϵ ist und somit nicht kleiner null sein kann.
Die Verteilung ist außerdem additiv, da die Anzahl molekularer Formen in den Erregungszuständen der Energien ϵ_1 und ϵ_2 durch $F(\epsilon_1) + F(\epsilon_2)$ gegeben ist.

Dieses Vorwissen wäre die Grundlage für die Anwendung der Maximum-Entropie-Methode.
Der Bayes'sche Ansatz beschränkt sich nicht auf dieses Wissen, wodurch er einen Vorteil gegenüber der *MaxEnt* hat. Mit der Wahl eines Modelltyps, der die Form der Peaks in einem Spektrum widerspiegeln soll, wird das Vorwissen bei der Bayes'schen Datenanalyse ergänzt.
Dies drücken Sivia und Carlile mit Hilfe des Theorems von Bayes (*vorgestellt in Abschnitt 2.2.1*) durch eine Faltung der Prior-Wahrscheinlichkeit $P(F|I)$ für das Spektrum F mit der Likelihood-Funktion $P(D|F,I)$ aus:

$$P(F|D,I) \propto P(D|F,I) \cdot P(F|I).$$

Die Relativierung zum Prior der Daten $P(D|I)$, der als Normierungskonstante gilt, lassen sie in ihren Überlegungen zur Vereinfachung weg.
Die Likelihood-Funktion beinhaltet Details über den Aufbau des Experiments sowie den gewählten Modelltyp in Form eines idealen Datensatzes $\hat{D}$.
Sivia und Carlile gehen für ihre Modellierung davon aus, dass die Daten durch

$$\hat{D}(\epsilon_k) = \int_{-\infty}^{+\infty} R(\epsilon_k - x)F(x)dx + B(\epsilon_k)$$

gegeben sind.
Das Modellspektrum $F(x)$ wird dabei mit der Auflösungsfunktion $R(\epsilon)$ des Messapparates gefaltet und ein Untergrundsignal $B(\epsilon_k)$ hinzu addiert.
Sie nehmen die zwei Funktionen R und B durch gute Schätzwerte als gegeben an. Für den Fall, dass das Untergrundsignal nicht gegeben sei, weisen sie darauf hin, seine Parameter für die Berechnung die Posterior-Wahrscheinlichkeit der Peakanzahl zu marginalisieren (*siehe Abschnitt 2.2.2*), da sie in diesem Zusammenhang uninteressant sind.
Zur Berechnung der Likelihood-Funktion wird in dem Artikel vereinfachend angenommen, dass die Messungen sich nicht gegenseitig beeinflussen. Die Daten werden daher als unabhängig angesehen.
Des Weiteren gehen Sivia und Carlile für ihre Experimente davon aus, dass die Daten von einem additiven, gaußschen Rauschen mit dem quadratischen Fehler σ_k gestört werden. Sie erhalten dadurch die Likelihood-Funktion:

$$P(D|F,I) = \frac{\exp(-\chi^2/2)}{\prod_k \left[2\pi\sigma_k^2\right]^{1/2}} \qquad \text{mit} \qquad \chi^2 = \sum_k \frac{[\hat{D}(\epsilon_k) - D(\epsilon_k)]^2}{\sigma_k^2}. \tag{2.1}$$

Die Prior-Wahrscheinlichkeit $P(F|I)$ für das Spektrum stellen sie mit dem zuvor erwähnten Vorwissen über das Spektrum auf.
Sie kommen über Argumente aus verschiedenen Bereichen, die sie in ihrem Artikel kurz benennen (logische Folgerichtigkeit, Informations- und Kodierungstheorie) [16–21], zu dem Schluss, dass die Prior-Wahrscheinlichkeit einer solchen Verteilung entropischer Form $\exp(\alpha S)$ sein müsse [22]. Dabei sei S die verallgemeinerte Shannon-Jaynes-Entropie eines Spektrums.
Mit dieser Folgerung begründen sie die Anwendung der Maximum-Entropie-Methode für ihre abschließende Datenanalyse [22, 23].

Sivia und Carlile formen den Prior mit weiteren Annahmen, die sie über ein Spektrum der Molekülspektroskopie machen können. Das Spektrum besteht aus "einigen" diskreten Erregungen, wie sie aus der Physik des Problems schließen.
Da sie für ihre Experimente annehmen, dass alle Peaks die Form einer gaußförmige Kurve mit der Breite W haben, beschreiben sie ihr Spektrum mit:

$$F(\epsilon) = \sum_{j=1}^{einige} A_j \cdot \exp\left[-\frac{(\epsilon - \epsilon_j)^2}{2W^2}\right].$$

Auf diese Weise bilden sie die Prior-Wahrscheinlichkeit $P(F|I)$ über das Vorwissen bezüglich der Parameterwerte für die Amplituden A_j und deren Lagen ϵ_j, $P(\{A_j, \epsilon_j\}|I)$.
Sie merken an, dass aufgrund der zur Anzahl der Messwerte verhältnismäßig kleinen Anzahl von Schätzparametern die Likelihood-Funktion verlässlicher als jede andere Wahrscheinlichkeitsverteilungsfunktion sei, die kein so genaues Vorwissen beinhaltet.
Aus diesem Grund werten sie die Likelihood-Funktion als ausschlaggebenden Teil der Posterior-Berechnung des Spektrums.
Sie machen zusätzlich die Annahme, dass die Prior-Wahrscheinlichkeit gleichmäßig, $P(F|I)$ konstant, ist. Dadurch stimmt die maximale Posterior-Wahrscheinlichkeit mit der Lösung des Maximum-Likelihood überein.
Da die Likelihood-Funktion mit $\exp(-\chi^2/2)$ aufgestellt wurde, erhält man die beste Schätzung für das Spektrum durch den Parametersatz $\{A_j, \epsilon_j\}$, der χ^2 minimiert.

Um die Frage nach der Anzahl der Peaks in einem Spektrum zu lösen, leiten Sivia und Carlile über das Bayes-Theorem die Gleichung der Posterior-Wahrscheinlichkeit der Peakanzahl N her [21, 24, 25]:

$$P(N|D) = \frac{P(D|N) \cdot P(N)}{P(D)}.$$

Das Vorwissen I lassen sie bei ihren Ausführungen zur Vereinfachung weg.
Der Prior der Daten $P(D)$ wird, wie in der Gleichung für $P(F|D, I)$, ausgespart.
Die Prior-Wahrscheinlichkeit $P(N)$ nehmen sie für den betreffenden Bereich von "einigen", 0 bis etwa 20 Peaks, als gleichmäßig an.
Damit geht sie in die Normierungskonstante K ein. Die Gleichung wird so zu $P(N|D) = K \cdot P(D|N)$.

Über die Marginalisierung (*siehe Abschnitt 2.2.2*) erhalten sie eine gemeinsame Wahrscheinlichkeitsverteilung für die Daten und Modellparameter:

$$P(D|N) = \iint \cdots \int P(D, \{A_j, \epsilon_j\}|N) d^N A_j \; d^N \epsilon_j.$$

$P(D, \{A_j, \epsilon_j\}|N)$ teilen Sivia und Carlile über die Produktregel (*siehe Abschnitt 2.2*) in die Likelihood-Funktion $P(D|\{A_j, \epsilon_j\}, N)$ und den Prior $P(\{A_j, \epsilon_j\}|N)$ auf.
Am Ende haben sie die zu lösende Gleichung:

$$P(N|D) = \frac{const \cdot N!}{[(\epsilon_{max} - \epsilon_{min}) \cdot A_{max}]^N} \cdot \iint \cdots \int \exp(-\chi^2/2) \; d^N A_j \; d^N \epsilon_j.$$

Der Prior für die Amplituden und deren Lagen $P(\{A_j, \epsilon_j\}|N)$ ist darin mit $[(\epsilon_{max} - \epsilon_{min}) \cdot A_{max}]^{-N}$ als gleichmäßige Verteilung in den Definitionsbereichen $\epsilon_{min} \leqslant \epsilon_j \leqslant \epsilon_{max}$ und $0 \leqslant A_j \leqslant A_{max}$ vertreten. Für den ϵ-Wertebereich können die verwendeten Energiegrenzen der Messung übernommen werden. Die Amplituden seien alle positiv. Mit A_{max} versuchen Sivia und Carlile die maximale Auslenkung des Spektrums festzuhalten. Sie definieren dadurch einen Wert, der weder von einer einzelnen Peakamplitude noch von der Summe aller Amplituden der N Peaks im Spektrum überschritten wird $(A_1 + A_2 + \cdots + A_j \leqslant A_{max})$.
Außerhalb des Definitionsbereichs ist die Prior-Wahrscheinlichkeit null.
Sivia und Carlile räumen ein, dass ein gleichmäßiger Prior möglicherweise als nicht optimal angesehen werden kann. Als Alternative nennen sie den Jeffreys' Prior, dessen Logarithmus der Amplituden gleichmäßig ist.
Ihr gewählter Prior verkompliziere die nachfolgenden Berechnungen nicht zu sehr. Die Form der Prior-Wahrscheinlichkeitsverteilung beeinflusse die Analyse außerdem nicht entscheidend, da mit der Anzahl der Peaks nur ein Parameter mit einer im Verhältnis dazu großen Datenmenge geschätzt wird.

Die Likelihood-Funktion $P(D|\{A_j, \epsilon_j\}, N)$ nähern sie mit χ^2 an, das sie aus Gleichung (2.1) übertragen.
Sivia und Carlile wählen zur Lösung des multiplen Integrals in der Gleichung einen analytischen Ansatz. Sie gehen davon aus, dass die Likelihood-Funktion nur ein signifikantes Maximum besitzt. Um dieses bilden sie eine quadratische Taylorreihenentwicklung:

$$\exp(-\chi^2/2) = \exp(-\chi^2_{min}/2) \cdot \exp[-(\delta\epsilon_j \;\; \delta A_j)^T \cdot \nabla\nabla\chi^2 \cdot \delta\epsilon_j \;\; \delta A_j/4].$$

Dadurch erhalten sie:

$$P(N|D) = \frac{const \cdot N!}{[(\epsilon_{max} - \epsilon_{min}) \cdot A_{max}]^N} \cdot \frac{(4\pi)^N \cdot \exp(-\chi^2/2)}{[Det(\nabla\nabla\chi^2)]^{\frac{1}{2}}},$$

wobei $Det(\nabla\nabla\chi^2)$ die Determinante der Hessischen Matrix darstellt.
Mit dieser letzten Gleichung berechnen sie die Posterior-Wahrscheinlichkeiten verschiedener N, um die Peakanzahl zu finden, die durch die gegebenen Messdaten am besten belegt wird.

Auswertung

In ihrem Artikel beschreiben Sivia und Carlile einen Algorithmus, mit dem sie eine praktische Umsetzung für ihre theoretischen Ergebnisse vorstellen.
Sie machen verschiedene Annahmen, die die Rechengeschwindigkeit und Robustheit des Algorithmus verbessern sollen.
Zunächst gehen sie davon aus, dass ein lineares Untergrundsignal B gegeben sei durch $B(\epsilon) = b_1\epsilon + b_2$.
Die geräteeigene Auflösungsfunktion R sei bekannt und für eine möglicherweise folgende lineare Interpolation fein genug definiert. Außerdem nehmen sie für ihre Messungen an, dass R nicht von der Energie abhängt.
Für ihre Anwendung gehen Sivia und Carlile davon aus, dass alle Peaks die gleiche spezifische Breite W haben und in Form der Gaußkurve beschrieben werden können.
Dabei geben sie an, dass ebenso die Lorentzfunktion oder andere Peakformen als Modelltyp möglich wären. Die Wahl der Peakform wird allgemein mit Hilfe der Wahrscheinlichkeitstheorie aus den Messdaten bestimmt.
Über das Parseval-Theorem legen sie weiterhin fest:

$$A_{max} = \frac{\text{Datenintensität} - \text{Untergrund}}{\text{Intensität in der Auflösungsfunktion}}.$$

Die dafür benötigte Anfangsschätzung des Untergrundsignals ließe sich beispielsweise mit dem Durchschnittswert der Daten an den beiden Energiebereichsgrenzen ϵ_{min} und ϵ_{max} berechnen.
Die Intensitäten der Daten und die Auflösungsfunktion müssten für die Berechnung der Gesamtintensität A_{max} angegeben sein.

Der Algorithmus, den die beiden in ihrem Artikel vorschlagen, ist einfach und durch bei ihnen erfolgreiche Ergebnisse gestützt. Sivia und Carlile führten mit dem Algorithmus Experimente verschiedener Spektren durch und konnten die enthaltenen Peakzahlen scheinbar ermitteln.
Die Experimente werden im nächsten Abschnitt beschrieben.
Zu Beginn ist der lineare Untergrund über seine zwei Parameter (b_1, b_2) zu schätzen. Dieser kann bei geringer Größe gleich null sein. Wenn in den Messdaten ein Untergrund erkennbar ist, kann dessen Schätzung aus der Grafik erfolgen.
Mit der Funktion für B wird dann eine eindimensionale Suche nach der Amplitudenlage des ersten, am stärksten ausgeprägten Peaks innerhalb des definierten Bereichs $\epsilon_{min} \leqslant \epsilon \leqslant \epsilon_{max}$ durchgeführt. Über die gefundene Lage kann mit den Messdaten der Amplitudenwert A_1 festgestellt werden. Daraufhin werden die Parameter, sowohl des Untergrundes (b_1, b_2) als auch des Peaks (A_1, ϵ_1) gleichzeitig innerhalb der Definitionsgrenzen langsam verändert. Mit den abgeänderten Parametern und dem festgelegten Peakmodell wird die Posterior-Wahrscheinlichkeit P(1| D) berechnet, um das kleinste χ^2 und damit die passendsten Parameter zu erhalten.
Für die Verfeinerung der Parameter schlagen Sivia und Carlile vor, einen simplex-ähnlichen Algorithmus [26] gefolgt von einer Newton-Raphson-Verfeinerung [27] zu verwenden.
Beim nächsten Algorithmusdurchlauf mit der Suche nach der Lage der zweiten Amplitude bzw. dem zweiten Peak im Spektrum wird das gefundene ϵ_1 als fester Wert übernommen.
Die Parameter werden verfeinert und der Posterior, bei dem das χ^2 seinen minimalen Wert annimmt, als Ergebnis der Verfeinerung mit den dazugehörigen Parameterwerten festgehalten.
Der Algorithmus wird so oft mit wachsendem N und wachsender Parameterzahl durchlaufen bis das eindeutige Maximum der $P(N|D)$-Werte erreicht wurde.

Durchgeführte Experimente

Für ihre Experimente verwenden Sivia und Carlile feste Werte für die spezifische Breite W der Peaks, da sie damit einen stabileren Programmablauf gewährleisten können. Sie lassen ihr Programm mehrere Male mit unterschiedlichen geschätzten Peakbreiten für ein Spektrum laufen.
In den Ergebnissen ist die Posterior-Wahrscheinlichkeit dadurch abhängig von der Peakanzahl und -breite W angegeben. Um die gewünschte Wahrscheinlichkeitsverteilung $P(N|D)$ zu erhalten, integrieren sie über W.
In ihrem Artikel beschreiben Sivia und Carlile Beispiele ihrer Experimente mit simulierten und real gemessenen Daten.
Die simulierten Daten erstellen sie durch die Faltung eines Spektrums einiger gaußförmiger Peaks mit einer gaußschen Auflösungsfunktion. Diesem künstlichen Signal fügen sie noch einen linearen Untergrund und Rauschen hinzu.
Mit Hilfe ihres Algorithmus und einer Schätzung der Peakbreite können sie die Peakanzahl ihres Spektrums plausibel bestimmen. Ihre gefundenen Amplitudenlagen weichen minimal ($\sim$ 0,2 %) von den wahren Werten ab, während die Amplituden auf 5 % genau festgestellt werden.
Um die Ergebnisse zu veranschaulichen und die Form der Posterior-Verteilung hervorzuheben, bilden sie den Logarithmus der $P(N|D)$-Wahrscheinlichkeiten und plotten diesen in einem zweidimensionalen Graphen mit den Peakzahlen.
Vergleichend berechnen sie die Wahrscheinlichkeitsverteilung eines dreimal so stark verrauschten Signals, wobei sich die maximale Wahrscheinlichkeit um einen Peak weniger verschiebt.
Diese Abweichung vom "wahren" Ergebnis beheben sie in einem dritten Experimentdurchlauf mit dem exakten Vorwissen über die Peakbreite des gaußförmigen Signals. Sie bessern dadurch die schlechte Datenqualität mit genauerem Wissen über die Spektrumsform aus.

In ihrem Artikel weisen sie auf die charakteristische Form des Lösungsgraphen für diese Art der Bayes-Analyse hin. Bis zum Maximum steigt die Wahrscheinlichkeit in ihren Experimenten mit simulierten Daten steil mit wachsender Peakanzahl, da das Modell nicht genug Struktur beinhaltet, um sich den Daten anzupassen. Nach dem Erreichen der maximalen Wahrscheinlichkeit fällt die Kurve aufgrund von komplexeren, aber nicht so gut passenden Modellen langsamer ab.
Sivia und Carlile erwähnen in diesem Zusammenhang die quantitative Aussage von Ockhams Rasiermesser. Diese besagt, dass beim Vorhandensein mehrerer Theorien über ein Phänomen die einfachste zu bevorzugen ist. In ihrem Fall ist dies die niedrigste Peakanzahl, die am besten zu den Daten passt.

Die realen Messwerte, die Sivia und Carlile für ihren Artikel verwenden, sind von einer 2,6-Dimethylpyridin- bzw. 2,6-Lutidin-Probe, ermittelt durch ein hoch-auflösendes IRIS-Spektrometer. Die Daten wurden zuvor von Mukhopadhyay et al. [28] mit der Maximum-Entropie-Methode analysiert.
Die Ergebnisse beider Methoden sind übereinstimmend, wobei Sivia und Carlile ihre Ergebnisdarstellung als klarer und zielgerichteter bezeichnen. Dies erklären sie mit dem umfangreicheren Vorwissen, das in die Bayes-Analyse eingeht.

Mit Hilfe der Wahrscheinlichkeitstheorie und ihrem Algorithmus können Sivia und Carlile die Frage nach der Anzahl der Peaks in einem Spektrum quantitativ beantworten.
Sie weisen dabei darauf hin, dass die Ergebnisse von der Gültigkeit ihrer vereinfachenden Annahmen abhängig sind und systematischen Ungenauigkeiten unterliegen.

2.4. Monte-Carlo-Verfahren zur Lösung von Integralen

Die Methode folgt einem numerischen Ansatz, um stochastische Problemstellungen zu lösen.
Dafür werden auf Grundlage der Wahrscheinlichkeitstheorie Zufallsexperimente durchgeführt.
Eine bedeutende Anwendung findet die Monte-Carlo-Methode in der numerischen Integration.
Vorteile hat sie gegenüber anderen numerischen Verfahren bei der Berechnung multidimensionaler Integrale. Auch mit komplexen Integrationsgrenzen kann sie mühelos umgehen.

Um ein Integral mit dem Monte-Carlo-Verfahren zu berechnen, werden zunächst
m Zufallspunkte $\vec{x} = (x_1, x_2, \ldots, x_m)$ erstellt. Der Vektor $\vec{x}$ beinhaltet Zufallswerte für die einzelnen Variablen, über die integriert werden soll.
Diese befinden sich innerhalb der definierten Integrationsgrenzen der jeweiligen Variablen. Die zufälligen Stützstellen werden zudem so erzeugt, dass sie gleichmäßig über den Definitionsbereich Ω verteilt sind.
Die Integrand-Funktion wird an den m Stützstellen ausgewertet.
Um eine Schätzung des zu lösenden Integrals zu erhalten, wird der Mittelwert dieser Funktionswerte mit dem Integrationsvolumen V multipliziert:

$$I = \int\int\cdots\int_{\Omega} f(\vec{x})\ d^n\vec{x} \qquad \rightarrow \qquad \hat{I} = \frac{V}{m}\sum_{i=1}^{m} f(\vec{x}_i)$$

$n :=$ Anzahl der Integrationsvariablen
$\vec{x}_i :=$ unabhängige, in Ω gleichverteilte Punkte

$$\sigma^2 = \frac{V}{m}\left(\int_V f^2(\vec{x})\ d\vec{x} - \left(\int_V f(\vec{x})\ d\vec{x}\right)^2\right)$$

Gemäß dem Gesetz der großen Zahlen und dem Bernoulli-Theorem [29] konvergieren der Schätzwert und der wahre Integrationswert mit wachsender Anzahl der Stützstellen. Der Fehler nimmt um $1/\sqrt{m}$ ab.
Eine weitere Möglichkeit die Varianz des geschätzten Integrals zu reduzieren, liegt in der Wahl der Stützstellen, d.h. Zufallsstichproben. Zwei Vorgehensweisen hierfür sind das Importance und das Stratified Sampling.

2.4.1. Importance Sampling

Um die Verteilung der Stützstellen auf "wichtige" Bereiche des Integrationsvolumens zu fokussieren, kann das Importance Sampling verwendet werden.
Die Idee dabei ist den Integrand mit einer Gewichtungsfunktion zu ergänzen:

$$\int_V f\,\overbrace{dV} = \int_V h \cdot \underbrace{g\ dV}, \qquad h = \frac{f}{g}$$

gleichmäßige Wahrscheinlichkeitsdichte
nicht gleichmäßige Wahrscheinlichkeitsdichte

Die Funktion g gibt dabei die Dichte der Stützstellenverteilung an.
Diese ist für das Integrationsvolumen normiert:

$$\int_V g(\vec{x})\ d\vec{x} = 1.$$

Die Varianz der Integralschätzung wird kleiner, wenn Bereiche großer Integrandwerte mittels g stärker gewichtet werden. Die Stützstellen werden gemäß der durch g gegebenen Gewichtung im Integrationsvolumen verteilt. Dadurch konzentrieren sich die Stützstellen auf Bereiche, in denen der

Integrand seine größten Werte aufweist.
Der Schätzer des Integrals unter Verwendung des Importance Samplings ist dann:

$$\hat{I}_{IS} = \frac{V}{m} \sum_{i=1}^{m} \frac{f(\vec{x}_i)}{g(\vec{x}_i)}$$

mit

$$\begin{aligned} \sigma_{IS}^2 &= \frac{V}{m} \left(\int_V \frac{f(\vec{x})^2}{g(\vec{x})^2} \cdot g(\vec{x}) \;\; d\vec{x} - \left(\int_V \frac{f(\vec{x})}{g(\vec{x})} \cdot g(\vec{x}) \;\; d\vec{x} \right)^2 \right) \\ &= \frac{V}{m} \left(\int_V \frac{f(\vec{x})^2}{g(\vec{x})} \quad d\vec{x} - \left(\int_V f(\vec{x}) \quad d\vec{x} \right)^2 \right). \end{aligned} \tag{2.2}$$

Es gibt verschiedene Wege die geeignete Dichtefunktion g zu wählen.
Sie sollte der zu integrierenden Funktion ähnlich sein, was ein bestimmtes Wissen über f voraussetzt. Außerdem sollte sie einfach abzutasten sein.
Wenn g die Funktion f ausreichend annähert, ist der neue Integrand f/g nahe 1 und dadurch die Varianz kleiner.
Eine Möglichkeit für den Fall, dass die zu integrierende Funktion einen sichtbaren Maximalwert bei $x = x_0$ besitzt, ist die Dichtefunktion durch eine Gaußsche Gewichtung zu definieren:

$$g(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left[-\frac{(x - x_0)^2}{2\sigma^2} \right]$$

Über die Varianz σ^2 ist die Breite der Gaußkurve nach Bedarf einstellbar.
Die Dichtefunktion, mit der das Importance Sampling die Varianz des Integrals verringert, ist die, die dem Integrand f am ähnlichsten ist und sie vorzugsweise einschließt.

2.4.2. Stratified Sampling

Bei dem Ansatz "geschichteter" Stützstellen wird das Integrationsvolumen in n Untervolumen, "Schichten", eingeteilt.
Um die Varianz der Integration mit dieser Methode zu verringern, werden in Bereichen großer Varianz des Integranden mehr Stützstellen gesetzt.
Dies kann durch unterschiedlich große Untervolumen mit gleicher Anzahl von Stützstellen umgesetzt werden. In dem Fall ist der Beitrag jedes Untervolumens an der Gesamtvarianz gleich groß mit σ^2/n.
Die Stützstellen werden durch kleinere Untervolumen in Bereichen großer Varianz konzentriert. Somit reduziert sich die Gesamtvarianz der Integration.
Eine weitere Möglichkeit besteht darin, die Anzahl der Stützstellen in einem Untervolumen proportional zur Varianz der Integrand-Funktion innerhalb des betreffenden Untervolumens zu wählen.
Die Untervolumen sind dabei alle gleich groß.

2.5. Der VEGAS-Algorithmus

Der Algorithmus wurde 1976 von G. P. Lepage entwickelt [30]. Er setzte dabei die Monte-Carlo-Methode in einem anpassungsfähigen Integrationsverfahren um.
Als grundlegenden Teil verwendet er in seinem Algorithmus das Importance Sampling (*siehe Abschnitt 2.4.1*). Durch eine geschickte Anwendung dieses Verfahrens eignet sich der VEGAS-Algorithmus vor allem für die Berechnung multidimensionaler Integrale.
Ein großes Problem bei der Integration über mehrere Dimensionen ist, dass das Integrationsvolumen

mit zunehmender Dimension exponentiell wächst. In anderen Integrationsmethoden werden K Werte entlang jeder Dimensions-Achse mit der Integrand-Funktion ausgewertet. Dadurch kommt es zu einer Berechnung von K^d (d:= Anzahl der Dimensionen) Werten.
Der Algorithmus von Lepage verwendet eine multidimensionale Gewichtungsfunktion für das Importance Sampling, die in die d Dimensionen teilbar ist:

$$p \propto g(x, y, z, \dots) = g_x(x) \cdot g_y(y) \cdot g_z(z) \dots .$$

Die Integration kann so über d getrennte eindimensionale Funktionsberechnungen, die jeweils aus K Werten bestehen, erfolgen. Die Werte stellen dabei Teilintegrale der K Untervolumen dar, in die das Integrationsvolumen geteilt wird. Damit wachsen die Anforderungen an den Algorithmus mit der Dimension nur linear ($\rightarrow K \cdot d$). Ebenso kann die passende Dichtefunktion p gefunden werden, indem ihre eindimensionalen Gewichtungsfunktionen g_i ($i = 1, \dots, d$) nacheinander ermittelt werden.
Aus diesen Gründen ist der VEGAS-Algorithmus für die Berechnung von Integralen hoher Dimensionen (> 4) besonders effizient.
Wie bei allen Methoden, die das Monte-Carlo-Verfahren zur Integration anwenden, muss der Integrand nicht stetig sein. So ist auch die Integration über Hypervolumina unregelmäßiger Form kein Problem für den Algorithmus.
Eine weitere nützliche Eigenschaft der Monte-Carlo-Anwendung ist, dass die Geschwindigkeit, mit der das Verfahren konvergiert, unabhängig von der Integraldimension ist.

Im Folgenden möchte ich auf Lepages Umsetzung des Algorithmus eingehen.
Neben den Integrationsgrenzen und der zu integrierenden Funktion benötigt der VEGAS-Algorithmus zu Beginn ein paar weitere Eingabeparameter.
Der Benutzer gibt dem Algorithmus mit *ncall* eine Vorgabe, wie viele Stützstellen für die Integralbestimmung gewünscht sind. Zu Beginn des Algorithmus wird mit dieser Angabe die tatsächliche Anzahl der zu erstellenden Stützstellen ermittelt. Dieser neu berechnete Wert wird auf die Variable *ncall* gesetzt.
Er ergibt sich aus der Anzahl der verwendeten Hypercubes und der Anzahl der Stützstellen pro Hypercube. Diese Werte werden beide ebenfalls durch den Algorithmus errechnet. Die Hypercubes sind Untervolumen, die sich durch ein erstelltes, mehrdimensionales Raster bilden. Die Rasterdimensionen entsprechen denen des Integrals, so dass es das Integrationsvolumen entlang jeder Dimensions-Achse einteilt.
Mit *itmx* wird die Anzahl der Iterationen vorgegeben, die der Algorithmus durchlaufen soll. Jede Iteration wertet die berechnete Anzahl Stützstellen mit der Integrand-Funktion aus. Eine wichtige Eingabe, um den VEGAS-Algorithmus zu steuern, ist der *init*-Wert. Mit ihm bestimmt der Benutzer, wie der Algorithmus vorgehen soll.
Für den Wert

init = 0 beginnt der Algorithmus mit einem gleichmäßigen Raster, das an den Integranden nach und nach angepasst wird. Es bedeutet einen Neustart des Algorithmus.

init = 1 Das angepasste Raster aus vorherigen Aufrufen des Algorithmus wird übernommen. Sonstige Ergebnisse aus den aufgerufenen Algorithmus-Durchläufen werden verworfen. Auf diese Weise gehen in die Ergebnisse dieses Durchlaufs nur Integralauswertungen aus diesem VEGAS-Aufruf ein.

init = 2 Das zuvor angepasste Raster sowie alle anderen Ergebnisse aus der Rasterfindung werden zur Berechnung des Integrals verwendet.

init = 3 Es werden keine Werte aus vorherigen Aufrufen übernommen. Mit dieser Aufrufvariante können Ergebnisse jeder einzelnen Iteration untersucht werden.

Der letzte Eingabeparameter ist *nprn*. Mit ihm lässt sich die Ausgabe kontrollieren.

Es kann bestimmt werden, ob neben den Ergebniswerten der Integration auch die Werte des dafür verwendeten Rasters ausgegeben werden sollen. Ebenso kann durch den Parameter angegeben werden, dass gar keine Ausgabe an den Benutzer erfolgen soll.

Über die Variable *mds* kann innerhalb des Algorithmus festgesetzt werden, ob das Stratified Sampling, wenn möglich, angewandt werden soll oder nur das Importance Sampling. In höheren Dimensionen ist die Unterteilung des Integrationsvolumens und die Berechnung der Varianzen der einzelnen Unterteilungen sehr aufwendig. Der Algorithmus prüft daher zu Beginn über die Anzahl der Stützstellen und der Dimensionen im Verhältnis zu den maximalen Unterteilungen pro Achse, ob das Stratified Sampling durchgeführt wird.
Das Importance Sampling wird für die Integration immer angewendet.
Jeder Hypercube beinhaltet gleich viele Stützstellen. Die Größe der Unterteilungen bzw. der Hypercubes richtet sich nach den Werten des Integranden innerhalb der entsprechenden Bereiche. So wird über die Hypercube-Größe die Dichte der Stützstellen reguliert.

Mit der am Anfang des Algorithmus stehenden Konstante *ALPHA* wird der Umfang angegeben, mit dem das zu Beginn gleichmäßige Raster, verändert werden soll. Das heißt in welchem Maße es sich von einer Iteration zur nächsten an den Integranden anpassen soll. Durch den Wert kann die Rasteranpassung beschleunigt, verlangsamt oder auch eingestellt (*ALPHA = 0*) werden.
Typischerweise liegt er zwischen 0,2 und 2 [30].

Eine weitere Konstante des Algorithmus ist *NDMX*. Über sie wird im Algorithmus die maximale Anzahl der Rasterunterteilungen pro Achse festgelegt. Ihr Defaultwert ist 50. Größere Werte ergeben keine signifikanten Gewinne in der Genauigkeit der Integrationsergebnisse [31].
Die maximale Dimension ist über die Konstante *MXDIM* angegeben. Mit ihr werden Arrays für verschiedene Variablen initialisiert. Der Default-Wert der maximalen Dimension im VEGAS-Code ist 10.

Der Integrand wird an *ncall* zufällig gewählten Stützstellen im definierten Volumen berechnet. Aus diesen Werten erstellt der Algorithmus ein gewichtetes Mittel, das eine Schätzung des gesuchten Integrals wiedergibt:

$$I \sim \hat{I} = \frac{1}{ncall} \sum_{j=0}^{ncall} \frac{f(\vec{x}_j)}{p(\vec{x}_j)}$$

$\vec{x} :=$ Stützstelle, $p :=$ Dichte der Punkte im Volumen

Der Algorithmus führt *itmx* statistisch unabhängige Berechnungen des gewünschten Integrals durch. Jede dieser Algorithmus-Iterationen beinhaltet *ncall* Funktionsauswertungen und hat als Ergebnis den Schätzwert $\hat{I}_i$. Dazu gehört jeweils die angenäherte Unsicherheit σ_i des geschätzten Integrals:

$$\sigma_i^2 = \frac{1}{ncall} \sum_{j=1}^{ncall} \frac{f^2(\vec{x}_j)}{p(\vec{x}_j)} - \hat{I}_i^2$$

Das Integrationsvolumen wird mittels eines multidimensionalen, rechteckigen Rasters in Hypercubes eingeteilt. Am Anfang des VEGAS-Algorithmus ist dieses gleichmäßig. Ein Beispiel eines solchen Rasters ist in Abbildung 2.4 für zwei Dimensionen zu sehen.

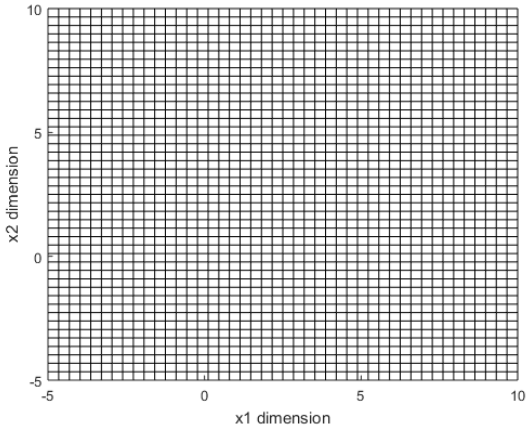


Abbildung 2.4.: Gleichmäßiges 2D-Anfangsraster

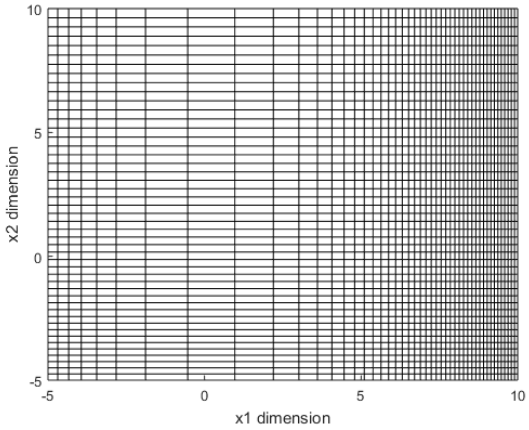


Abbildung 2.5.: Verfeinertes zweidimensionales Raster zu einer Integrand-Funktion mit Extremwerten an den Integralsgrenzen der Abszisse. In Richtung der x2-Achse ist keine eindeutige Verfeinerung zu erkennen, da die Extremwerte dieser Koordinate außerhalb des Integrationsvolumens liegen.

Der VEGAS-Algorithmus passt nach jeder Iteration die Dichte $p(x)$ mittels der neugewonnenen Information über den Integranden an.
Die Anzahl der Stützstellen bleibt in den einzelnen Hypercubes einheitlich. Die Dichte wird verfeinert, indem die Größe der Würfel auf die darin enthaltenen Funktionswerte abgestimmt wird. Die Hypercubes werden kleiner, je größer die Funktionswerte darin sind.
Durch dieses Vorgehen sind mehr Stützstellen in den Bereichen mit großen Funktionswerten vorhanden und die Varianz verringert sich mit jeder Iteration. Die schrittweise Anpassung des Rasters ist vollendet, wenn zu erkennen ist, dass die Varianz sich nicht mehr merklich verringert.
An diesem Punkt ist das optimale Raster erreicht und das Importance Sampling vollständig umgesetzt. In Abbildung 2.5 ist ein durch den Algorithmus ermitteltes Raster veranschaulicht. Das Abändern des Rasters übernimmt die *rebin*-Methode, die vom VEGAS-Algorithmus aus aufgerufen wird.

Über den *init*-Wert 0 ist es möglich vor der endgültigen Integralberechnung mehrere Iterationen durchzuführen, mit denen das Raster optimiert wird. Dazu ist eine moderate Anzahl von Stützstellen ausreichend [30].
Für den darauffolgenden VEGAS-Aufruf zur genaueren Berechnung des Integrals mit dem zuvor gefundenen Raster sollte die Stützstellenanzahl erhöht werden.

Die Ergebnisse der durchlaufenen Iterationen werden kombiniert zum besten Integral (I_{best}), dem dazugehörenden geschätzten Fehler (σ_{best}) und dem Grad der Verlässlichkeit des Ergebnisses (χ^2) pro Freiheitsgrad (m):

$$I_{best} = \sum_{i=0}^{m-1} \frac{\hat{I}_i}{\sigma_i^2} \Bigg/ \sum_{i=0}^{m-1} \frac{1}{\sigma_i^2} \qquad \sigma_{best} = \left(\sum_{i=0}^{m-1} \frac{1}{\sigma_i^2} \right)^{-\frac{1}{2}}$$

$$\frac{\chi^2}{m} = \frac{1}{m-1} \sum_{i=0}^{m-1} \frac{(\hat{I}_i - I_{best})^2}{\sigma_i^2}$$

$m :=$ Anzahl der durchgeführten Iterationen

Die Verwendung von χ^2 beruht auf der Voraussetzung, dass die Schätzer $\hat{I}_i$ normalverteilt um den wahren Wert I liegen. Diese Bedingung kann über das zentrale Grenzwerttheorem erfüllt werden, indem *ncall* groß genug gewählt und $f(x)$ quadratisch integrierbar ist [31].
I_{best} kann als verlässlicher Schätzer betrachtet werden, wenn χ^2 pro Freiheitsgrad nicht größer als eins ist.
Die Anzahl der Stützstellen, *ncall*, ist abhängig von der zu integrierenden Funktion. Das geeignetste *ncall* für einen zuverlässigen Integralschätzer ist experimentell zu bestimmen.

2.5.1. Aktuellste VEGAS-Version in Python

Seit Juli 2015 gibt es eine Version des VEGAS-Algorithmus in Python, die G.P. Lepage weiterentwickelt hat. Der ursprüngliche Algorithmus ist darin leicht verändert.
Die Anzahl der Stützstellen pro Hypercube passt sich nun der berechneten Standardabweichung in dem jeweiligen Untervolumen an. Auf diese Weise hat Lepage das Stratified Sampling auch in höheren Dimensionen effizient umgesetzt. Bei der vorherigen Version seines Algorithmus merkte er an, dass das Stratified Sampling noch keine Vorteile in hohen Dimensionen bringe [30]. Daher kam es in der Version nur in niedrigen Dimensionen zur Anwendung.
Einige neue Eingabeparameter geben dem Anwender Einfluss auf den Ablauf des Algorithmus. Es gibt zum Beispiel die Möglichkeit die Stützstellen in Stapeln (Batches) der Integrand-Funktion zu übergeben und gleichzeitig auswerten zu lassen. Die Größe dieser Stapel ist variierbar.

3. Methoden

Wie Sivia und Carlile wende ich den Bayes'schen Ansatz und die Wahrscheinlichkeitstheorie zur Auswertung eines NMR-Spektrums an.
Zunächst leite ich die Gleichung der Wahrscheinlichkeitsverteilung für die Anzahl der Peaks in einem NMR-Spektrum her. Innerhalb dieser Gleichung entsteht durch Marginalisierung einiger Parameter ein multidimensionales Integral.
Ich benenne unterschiedliche Möglichkeiten, ein solches Integral zu lösen. Die Umsetzung der angedachten Methode für diese Arbeit beschreibe ich daraufhin genauer.
Abschließend erläutere ich meine Realisierung, um die Anzahl der Peaks in einem Spektrum zu bestimmen, auch im Vergleich zur Vorgehensweise von Sivia und Carlile.

3.1. Gleichung zur Berechnung der Anzahl der Peaks in einem Spektrum

Vor der Messung ist über das Spektrum F ein bestimmtes Wissen I bekannt, das auch den Versuchsaufbau beinhaltet.
Durch die Abtastung des Signals und anschließende Fouriertransformation ergeben sich M gemessene Werte. Diese werden in Form eines Datensatzes $\{D(x_k)\}$ mit $k = 1, 2, 3, \ldots, M$ festgehalten.
An dem Graphen von Messpunkten einer Probe eines reinen Stoffes ist ersichtlich, dass das transformierte Signal durch die Kurve einer nicht zu komplexen Funktion angenähert werden kann.
Für allgemeine Analysen von Proben, in denen mehr als ein Stoff zu vermuten ist, wird versucht, das Spektrum mit mehreren solcher Kurven nachzubilden. Ich lege für meine Überlegungen fest, dass sich ein Spektrum $F(x)$ aus N Peaks zusammensetzt.
Über den Verlauf eines Peaks nehme ich an, dass er durch eine Lorentzfunktion beschrieben werden kann.

Lorentzfunktion (aus dem Pseudo-Voigt-Profil [32])

$$L(x, A, x_0, \Gamma) = \frac{A}{1 + \left(\frac{x - x_0}{\Gamma/2}\right)^2} \tag{3.1}$$

$A :=$ Amplitude,
$x_0 :=$ x-Achsenabschnitt, an dem die Funktion ihren maximalen Wert A annimmt,
$\Gamma :=$ Halbwertsbreite

Das gemessene NMR-Signal einer Stoffprobe führt mit dem FID eine gedämpfte Schwingung aus (*siehe Abschnitt 2.1.1*). Der Abfall der Amplituden verläuft dabei exponentiell. Aus diesem Grund ist die umhüllende Funktion des FID durch eine Exponentialfunktion darstellbar.
Über die Fouriertransformation erhält man aus dem Realteil des FID ein Frequenzspektrum. Da die Fouriertransformierte einer Exponentialkurve eine Lorentzkurve ist, kann davon ausgegangen werden, dass diese die Form eines Peaks passend wiedergibt.
Ich erstelle meine Arbeit in Bezug auf die NMR-Spektroskopie. Bei dieser zeigt die Abszisse eines Spektrumsgraphen die chemische Verschiebung δ an. Daher bezeichne ich die Lage der Amplitude in meiner Arbeit als δ_0.
Die Halbwertsbreite nenne ich vereinfachend w (von engl. width).

Ich kennzeichne diese beiden Größen sowie die Amplitude A als Modellparameter für die zu analysierenden Daten.
Um die Anzahl N der im Spektrum enthaltenen Peaks zu bestimmen, berechne ich die Wahrscheinlichkeit $P(N|D,I)$ für unterschiedliche N. Die beste Schätzung der Peakanzahl ergibt sich aus der maximalen Posterior-Wahrscheinlichkeit.
Zur Berechnung des Posterior-Wertes nutze ich das Bayes-Theorem (*siehe Abschnitt 2.2.1*):

$$P(N|D,I) = \frac{P(D|N,I) \times P(N|I)}{P(D|I)}.$$

Die Prior-Wahrscheinlichkeit für die Daten $P(D|I)$ ist für alle möglichen Werte der Peakanzahl N gleich. Sie verändert alle Posterior-Wahrscheinlichkeiten gleichermaßen.
Daher hat der Prior der Daten keinen Einfluss auf das Ermitteln des Posterior-Maximums. Mit oder ohne $P(D|I)$ bleibt das Maximum beim Posterior der gleichen Peakanzahl.
Der Prior-Wert für die Daten kann aus diesem Grund als Normierungskonstante der Posterior-Wahrscheinlichkeiten angesehen und in den Berechnungen vernachlässigt werden.
$P(N|I)$ ist die Prior-Wahrscheinlichkeit für eine bestimmte Anzahl von Peaks im Spektrum. Diese ist für relativ kleine N (0 - 20) gleichmäßig. Vor der Datenanalyse gelten alle N in diesem Rahmen von ein paar wenigen Peaks als gleichwahrscheinlich: $P(N|I) = 1/N_{max}$. Ist die höchste Peakanzahl, deren Posterior-Wahrscheinlichkeit berechnet wird, N_{max} beispielsweise 20, so ist der Prior aller Peakzahlen $1/20$. Ich behandle diesen Prior wie $P(D|I)$, weil er die maximale Wahrscheinlichkeit lediglich skaliert.
Die zu lösende Gleichung reduziert sich dadurch auf $P(N|D,I) \propto P(D|N,I)$.
Um die Likelihood-Funktion $P(D|N,I)$ berechnen zu können, ist Wissen über die Form der Daten notwendig. Daher nehme ich die zuvor identifizierten Modellparameter hinzu.
Das Modell erfordert N Parametersätze $\{A_j, \delta_{0_j}, w_j\}$ mit $j = 1, 2, 3, \ldots, N$.
Diese müsste ich schätzen.
Da die Werte der Modellparameter für die Bestimmung der Peakanzahl uninteressant sind, kann ich ihre Schätzung auslassen und sie stattdessen marginalisieren (*siehe Abschnitt 2.2.2*):

$$P(D|N,I) = \iint \cdots \int P(D, \{A_j, \delta_{0_j}, w_j\}|N,I) \;\; d^N A_j \;\; d^N \delta_{0_j} \;\; d^N w_j. \tag{3.2}$$

Über die Produktregel (*siehe Abschnitt 2.2*) erhalte ich die Gleichung für die zu integrierende Funktion

$$P(D, \{A_j, \delta_{0_j}, w_j\}|N,I) = \;\; P(D|\{A_j, \delta_{0_j}, w_j\}, N, I) \;\; \times \;\; P(\{A_j, \delta_{0_j}, w_j\}|N,I).$$

Die Prior-Wahrscheinlichkeit $P(\{A_j, \delta_{0_j}, w_j\}|N,I)$ für die Amplituden A_j, ihre Lagen δ_{0_j} und die Halbwertsbreiten w_j der Peaks nehme ich als gleichmäßig an. Für die Anwendung ist es sinnvoll, die Werte der Modellparameter durch jeweilige Minima und Maxima zu begrenzen. Die Werte innerhalb der Definitionsgrenzen sehe ich als gleichwahrscheinlich an. Außerhalb dieses Definitionsbereichs gilt der Prior gleich 0.
So kann die Prior-Wahrscheinlichkeit der Modellparameter beschrieben werden durch

$$P(\{A_j, \delta_{0_j}, w_j\}|N,I) = [(A_{max} - A_{min}) \cdot (\delta_{max} - \delta_{min}) \cdot (w_{max} - w_{min})]^{-N}.$$

Die Likelihood-Funktion $P(D|\{A_j, \delta_{0_j}, w_j\}, N, I)$ nähere ich an über

$$\left[\prod_{k=1}^{M} \sqrt{2\pi\sigma_k^2}\right]^{-1} \cdot \exp(-\chi^2/2)$$

mit der Summe der Quadrate der gewichteten Differenzen:

$$\chi^2 = \sum_{k}^{M} \frac{[\hat{D}(\delta_k) - D(\delta_k)]^2}{\sigma_k^2} \qquad , \;\; \hat{D}(\delta_k) := \text{idealer bzw. Modell-Datensatz.}$$

Der erste Faktor der Likelihood-Funktion beschreibt ein additives gaußsches Rauschen, das ich bei der Messung vermute. Ich nehme des Weiteren an, dass die gemessenen Daten gleichverteilt um die wahren Datenwerte liegen. Aus diesem Grund verwende ich χ^2 zur Annäherung.
Die Reihenfolge, in der die einzelnen Peaks eines Spektrums ermittelt werden, ist nicht festgelegt. Um dies in die Wahrscheinlichkeitsberechnung einzubringen, wird der Faktor $N!$, die Anzahl möglicher Indizierungen der einzelnen Peaks, Teil der Gleichung:

$$P(N|D,I) = \frac{c}{[(A_{max}-A_{min})\cdot(\delta_{max}-\delta_{min})\cdot(w_{max}-w_{min})]^N} \times \iint\cdots\int \exp\left[-\frac{1}{2}\left(\sum_k^M \frac{[\hat{D}(\delta_k)-D(\delta_k)]^2}{\sigma_k^2}\right)\right] d^N A_j \;\; d^N\delta_{0_j} \;\; d^N w_j \tag{3.3}$$

$$\text{mit} \quad c = \frac{N!}{\prod_{k=1}^{M}\sqrt{2\pi\sigma_k^2}}.$$

Für meine Arbeit nehme ich eine einheitliche Standardabweichung σ_k aller Messwerte k an. Dadurch wird jeder Messwert gleich gewichtet. Das bedeutet, dass ich alle Messdaten als gleich wichtig betrachte. Außerdem wird σ_k damit zu einem konstanten Wert, der das Ergebnis nur skaliert.
Ebenso verhält es sich mit dem Term $\sqrt{2\pi}^M$, der für jede Wahrscheinlichkeit $P(N|D,I)$ identisch ist.
Der Varianz der Messwerte σ_k^2, die die χ^2-Annäherung skaliert, weise ich als beliebig wählbare Konstante den Wert 1 zu.
Allgemein werden ideale Daten durch eine Faltung der wahren Daten mit der Auflösungsfunktion R des Messapparates und dazu addiertem Hintergrund B angegeben, wie in Abschnitt 2.3 erwähnt:

$$\hat{D}(\delta_k) = \int_{-\infty}^{\infty} R(\delta_k-\delta)F(\delta) \;\; d\delta + B(\delta_k).$$

In meiner Arbeit gehe ich davon aus, dass das Untergrundsignal durch eine geeignete Phasenkorrektur nach der Messung beseitigt wurde.
Die Auflösungsfunktion ist in der Modellierung beinhaltet. Die Lorentzfunktion stellt die Faltung des wahren Signals mit der Auflösungsfunktion dar.

Mit den beschriebenen Annahmen kann die Wahrscheinlichkeit für die Anzahl der Peaks in einem Spektrum schließlich berechnet werden durch:

$$P(N|D,I) = \frac{N!}{[(A_{max}-A_{min})\cdot(\delta_{max}-\delta_{min})\cdot(w_{max}-w_{min})]^N} \times \iint\cdots\int \exp\left[-\frac{1}{2}\left(\sum_k^M [\hat{D}(\delta_k)-D(\delta_k)]^2\right)\right] d^N A_j \;\; d^N\delta_{0_j} \;\; d^N w_j \;\; , \tag{3.4}$$

wobei ich die idealen Datenwerte $\hat{D}(\delta_k)$ modelliere in der Form:

$$\sum_{j=1}^{N} \frac{A_j}{1+\left(\frac{(\delta_k-\delta_{0_j})}{w_j/2}\right)^2}.$$

3.2. Lösung des multiplen Integrals in $P(N|D,I)$

Innerhalb der Gleichung für die Posterior-Wahrscheinlichkeit der Peakanzahl aus dem vorausgehenden Abschnitt entsteht aus der Likelihood-Funktion $P(D|N,I)$ eine marginale Verteilung über die Wahrscheinlichkeitsverteilung der Daten und Modellparameter (*siehe Gleichung (3.2)*).
Das Integral erweitert sich mit wachsender Peakanzahl um drei Dimensionen. So umfasst das zu berechnende Integral $3N$ Dimensionen und stellt ein multiples Integral dar, das es zu lösen gilt.

Es gibt verschiedene Arten ein mehrdimensionales Integral zu berechnen.
Dabei kann zwischen numerischer und analytischer Integration unterschieden werden.
Bei der analytischen Herangehensweise wird Wissen über den funktionalen Zusammenhang, in dem das zu lösende Problem steht, in die Berechnung des Integrals einbezogen. Wie in Abschnitt 2.3 beschrieben, verwenden Sivia und Carlile eine analytische Vorgehensweise mit Hilfe der Taylorentwicklung.
Je nach Problemstellung kann die analytische Integration schwierig oder unmöglich sein.

Eine Möglichkeit der numerischen Integration ist das Erstellen einer Matrix mit der gleichen Dimensionsgröße wie das zu lösende Integral. Darin sind mögliche Wertekombinationen der einzelnen Parameter vertreten, die sich innerhalb ihrer Integralgrenzen befinden.
Die Schrittgröße, mit der der Wert eines Modellparameters variiert, gibt die gewählte Präzision bzw. den Stützstellenabstand in der entsprechenden Dimension wieder. Für jede Wertekombination wird der Funktionswert des Integranden ermittelt und damit Stützstellen für die Integration gebildet. Die Stützstellen sind mit der Schrittgröße der betreffenden Dimension gleichmäßig im Integrationsvolumen verteilt.
Die Matrix wird durch Aufsummieren entlang der einzelnen Dimensionen zusammengezogen, um den Integralwert zu erhalten.
In verschiedenen Programmiersprachen, wie Matlab und MuPAD, stehen Funktionen zur numerischen Integration bereit. MuPAD hat eine Funktion, um ein einfaches Integral zu berechnen. In Matlab gibt es drei Funktionen, die ein ein- bis drei-dimensionales Integral lösen. Um über mehr Dimensionen zu integrieren, können die Funktionen der jeweiligen Prorammiersprache geeignet verschachtelt werden.

Einen weiteren Lösungsweg stellt die in Abschnitt 2.4 beschriebene Monte-Carlo-Methode dar.
Sie wurde bereits in verschiedenen Algorithmen zur Berechnung mehrdimensionaler Integrale umgesetzt. Ein Beispiel dafür ist der in Abschnitt 2.5 vorgestellte VEGAS-Algorithmus von Lepage.
Ich habe ihn für die Auswertung gewählt, da er anpassungsfähig und performant ist. Laut Lepage [30] ist er verglichen mit anderen Algorithmen, die das Monte-Carlo-Verfahren oder ähnliche Techniken verwenden, effizienter in hohen Dimensionen.

3.2.1. Umsetzung des VEGAS-Algorithmus

Für meine Anwendung habe ich den VEGAS-Algorithmus in *C* aus den Numerical Recipes (NR) [27] herangezogen. Dieser stimmt im Wesentlichen mit der Standardversion des von Lepage in Fortran erstellten Algorithmus überein. Laut NR [33] haben sie bei der Übertragung in C-Code minimale Veränderungen vorgenommen, um den Code ihren Konventionen anzupassen.
Ich habe den von NR bereitgestellten Code für diese Arbeit in Matlab implementiert. Dafür habe ich zunächst die verwendeten Variablen eindeutiger identifiziert.
Die Dokumentation der Numerical Recipes wie auch vor allem zwei Veröffentlichungen von Lepage, "VEGAS - An Adaptive Multi-dimensional Integration Program" [31] und "A new algorithm for adaptive multidimensional integration" [30], waren dabei sehr hilfreich. In Anhang A befindet sich die Beschreibung der einzelnen Variablen, die im Algorithmus verwendet werden. Ich habe diese aus den genannten Dokumenten zusammengetragen und einige selbst aus dem Codezusammenhang erarbeitet.
In Matlab habe ich eine Datei namens *vegas.m* mit den Funktionen *vegas* und *rebin* erstellt.

Die beiden haben die gleiche Funktionsweise wie die in Abschnitt 2.5 beschriebenen, gleichnamigen Methoden des Original-Algorithmus. Ich habe den ursprünglichen Code aufgrund der normalen semantischen und Syntax-Unterschiede zwischen den Programmiersprachen an Matlab angepasst. Ebenfalls habe ich die verwendeten Indices in den Arrays und Operationen passend abgeändert[1]. Ein paar Modifikationen, die nicht gleich im Code als notwendig erkennbar waren, erläutere ich im folgenden Abschnitt etwas genauer.

Unterschiede zum ursprünglichen VEGAS-Code in C

Um den VEGAS-Algorithmus in Matlab zum Laufen zu bringen, habe ich die Zahlenarten der verschiedenen Rechenvariablen angepasst. Dies sind ganze und Fließkomma-Zahlen. In der Implementierung entsprechen diese den Typen *Integer (int)* und *Double*. In C geschieht die eindeutige Typzuweisung bei der Initialisierung der Variablen. Dabei muss noch kein Wert für die Variable gesetzt werden. Danach kann einer Variablen kein Wert, der nicht ihrem zugehörigen Typ entspricht, zugewiesen werden. Matlab benötigt keine eindeutige Typdeklaration für die Variablen. Eine nicht mit einem Datentyp belegte Variable erhält den Typ *double*. Es können ihr danach Werte unterschiedlicher Datentypen zugewiesen werden. Daher war es wichtig, in meinen Matlab-Code mit Typumwandlungen sicherzustellen, dass die Variablen die geeignete Zahlenart für die gewünschte Operation besitzen. Bei den Umwandlungen in ganze Zahlen habe ich $0,5$ von dem zugeordneten Wert abgezogen, um das selbe Ergebnis wie im Ursprungs-Code zu erhalten. In C wird die Zahl bei der Umwandlung zu Integer um die Ziffern nach dem Komma gekürzt. Matlab rundet dagegen kaufmännisch auf oder ab.

Um die korrekte Berechnung von Quotienten zu gewährleisten, habe ich die Dividenden und Divisoren als Fließkommazahlen angegeben. Der Ergebniswert wird andernfalls in Matlab als kaufmännisch gerundete Zahl ausgegeben.
Ich möchte dies an einem Zahlenbeispiel veranschaulichen:

C	Matlab
int $a = 3, b = 2, c;$ $c = a/b$ Ergebnis: $c = 1$	$a = 3; b = 2;$ $c = a/b = \text{double}(a)/\text{double}(b)$ Ergebnis: $c = 1.5$ $c = \text{int}(a/b) \quad \rightarrow \quad c = 2$
	$c = \text{int}(a)/\text{int}(b)$ $= \text{int}(a)/\text{double}(b)$ $= \text{double}(a)/\text{int}(b)$ Ergebnis: $c = 2$
	$c = \text{int}(\text{double}(a)/\text{double}(b) - 0.5)$ Ergebnis: $c = 1$

Die Genauigkeitsgrenze habe ich für die Berechnungen in Matlab angepasst. Der kleinste Wert, der in meiner Implementierung als Ergebnis akzeptiert wird, ist $1 \cdot 10^{-16}$. Im Code habe ich diesen durch die Konstante *TINY* festgelegt. Der Wert liegt am Rande der numerischen Rechengenauigkeit von Matlab. Dadurch ist gewährleistet, dass die Ergebniswerte nicht präziser angegeben werden, als es die Programmiersprache wirklich berechnen kann. Dies kann zum Beispiel bei Rechnungen mit

[1] In *C* ist der erste Index für Arrays/ Matrizen 0, in *Matlab* 1.

Zehnerpotenzen in Matlab der Fall sein.

In der ursprünglichen Version des VEGAS-Algorithmus gibt es als Parameter der Integrand-Funktion neben der Stützstelle *x* noch die Gewichtungsgröße *wgt*. Diese habe ich bei meiner Implementierung des Integranden ausgelassen. In NR wird darauf hingewiesen, dass *wgt* in der Integrand-Funktion meistens ignoriert wird. Sie kommt nur dann zum Einsatz, wenn über mehr als eine Funktion integriert werden soll. Dies ist für die Aufgaben meiner Arbeit nicht der Fall.
Eine weitere Anpassung an meine Anwendung, die nichts an der Funktionsweise von *vegas* ändert, ist das Hinzufügen eines Eingabeparameters.
Dieser ist ein *Flag*[2] für die Ausgabewerte der Ergebnisse. Es gibt an, ob die Ergebniswerte als Festkommazahl[3] oder in einer kompakteren Form mit Zehnerpotenzen angegeben werden sollen. Die Wahl dafür ist abhängig von der Integrand-Funktion bzw. der Größe des Integrals.
Im Original-VEGAS-Code gibt es, wie ich in Abschnitt 2.5 erwähnt habe, die Konstante *MXDIM*. Sie legt die maximale Dimension fest. Da ich den Algorithmus für unterschiedlich hohe Dimensionen verwende, habe ich diese Konstante nicht in meinen VEGAS-Code übernommen. Ich habe stattdessen die Integraldimension *ndim* gebraucht.
Meine Matlab-Implementierung des VEGAS-Algorithmus ist abgedruckt in Anhang B.1 und, wie alle erstellten Matlab-Dateien zu dieser Arbeit, auf der beiliegenden CD-Rom zu finden.

3.2.2. Implementierter Ablauf zur Integration mit dem VEGAS-Algorithmus

Für die Aufrufe der *vegas*-Funktion mit den geeigneten Parametern habe ich für die unterschiedlichen Anwendungen eigene Matlab-Funktionen in entsprechenden Dateien erstellt.
Ich habe bei allen diesen Funktionen einige allgemeine Regelungen implementiert, die ich hier kurz präsentieren möchte.
Wie in Abschnitt 2.5 erwähnt, ist der Fehler der *vegas*-Integration, bis das optimale Raster gefunden ist, groß. Die Varianz wird durch die Rasteranpassung verkleinert.
Für die Entwicklung des Rasters rufe ich daher zuerst die *vegas*-Funktion mit *init = 0* auf. Ich habe festgelegt, dass der Integrand darin fünf mal mit den *ncall* Stützstellen evaluiert wird.
Um sicherzustellen, dass ein optimales Raster gefunden wurde, überprüfe ich den χ^2 - Wert der letzten Iteration. Nur wenn der Wert kleiner als die Anzahl der Evaluierungs-Iterationen minus eins ist, gilt das Raster als zuverlässig.
Ich veranlasse bis zu fünf Aufrufe der *vegas*-Funktion, um ein zuverlässiges Raster zu finden. Die Anzahl benötigter *vegas*-Aufrufe zum Finden des optimalen Rasters wird dem Benutzer ausgegeben.
Den Wert der Konstante *ALPHA* für die Anpassungsrate des Rasters habe ich für alle meine Anwendungen auf $1,5$ gesetzt.
Mit dem gefundenen Raster führt meine Anwendung dann eine Iteration durch, die mit mehr Stützstellen ein genaueres Integrationsergebnis erzielt. Ich habe dafür die Vorgabe für die Anzahl der Stützstellen für den zweiten *vegas*-Aufruf zehnmal höher gewählt als für den ersten.
Aufgrund der Verwendung des Monte-Carlo-Verfahrens, das die Stützstellen der Integration mittels Zufallszahlen erstellt, schwanken die Ergebnisse.
Daher lasse ich den Algorithmus mehrmals mit den gleichen Eingabewerten laufen, um die Tendenz zu dem gesuchten, wahren Ergebnis erkennen zu können.
Für die Entwicklung der Anwendung und zum Testen der Implementierung habe ich eine Beispielanwendung erstellt. Diese beschreibe ich im folgenden Abschnitt.

[2] Statuskennzeichnung, besitzt meist einen Wert: 0 oder 1
[3] Bei Festkommazahlen ist die Anzahl der Ziffern vor und hinter dem Komma fix.

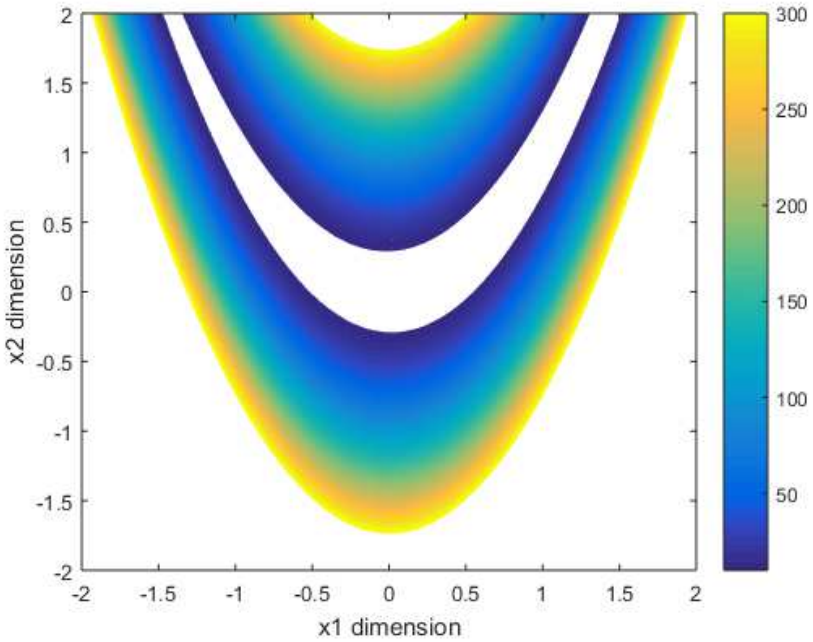


Abbildung 3.1.: 2D Rosenbrock-Funktion: Die Farbe gibt die Höhe des Funktionswertes an der entsprechenden Stelle (x1, x2) an.
(Die Kontur-Niveaus habe ich dabei auf Werte zwischen 10 und 300 skaliert.)

Beispielanwendung: Integration der Rosenbrock-/ Banana-Funktion

Sie dient normalerweise als Performanz-Testproblem für Optimierungsalgorithmen [34]. Ich habe sie als Testanwendung ausgewählt, da sie für beliebige Dimensionen verwendet werden kann. Zudem ist sie quadratisch integrierbar, was für die Verwendung des VEGAS-Algorithmus vorausgesetzt ist (siehe Abschnitt 2.5). Definiert ist sie durch:

$$f(\vec{x}) = \sum_{i=1}^{d-1} (1 - x_i)^2 + 100 \cdot \left(x_{i+1} - x_i^2\right)^2 \tag{3.5}$$

$d :=$ Anzahl der Dimensionen.

Das globale Minimum der Funktion, $f(\vec{x}^*) = 0$, liegt bei $\vec{x}^* = (1, \ldots, 1)$.
Es befindet sich in einem engen, parabolischen Tal, das an die Form einer Banane erinnert (siehe Abbildung 3.1). Deshalb wird die Funktion auch als "Banana"- Funktion bezeichnet.
Evaluiert wird sie gewöhnlich auf dem Hypercube $x_i \in [-5;\ 10]$ für alle $i = 1, 2, 3, \ldots, d$ [34].
Je nach Anwendung kann dieser Bereich, z.B. für Berechnungen mit höheren Dimensionen, auf $x_i \in [-2.048;\ 2.048]$ für alle $i = 1, 2, 3, \ldots, d$ begrenzt werden, um die Rechenzeit zu verringern.
Um den implementierten VEGAS-Algorithmus zu überprüfen, habe ich mit ihm die Integrale der zwei- und fünf-dimensionalen Rosenbrock-Funktion berechnet. Das Integrationsvolumen ist dabei über die genannten Evaluationsgrenzen definiert. Die Ergebnisse der VEGAS-Integration habe ich mit den wahren Integralen verglichen. Diese habe ich mit MuPad berechnet.
Die Matlab-Funktion dieser Anwendung ist in Anhang B.2 abgebildet. Über einen Eingabeparameter kann die gewünschte Dimension der Rosenbrock-Funktion gesetzt werden.

3.3. Berechnung der Anzahl der Peaks in einem Spektrum

Für die Anwendungen zur Bestimmung der Peakanzahl habe ich eine Variable *n_max* erstellt. Ihr Wert gibt die maximale Peakanzahl an, deren Posterior-Wahrscheinlicheit berechnet werden soll. Die Angabe von *n_max* sollte ein paar Peaks mehr sein als im Spektrum vermutet, um die größte Wahrscheinlichkeit in Abhängigkeit der Peakanzahl erkennen zu können. Für ein bis *n_max* Peaks errechnet meine Anwendung die Posterior-Wahrscheinlichkeiten und legt sie in Form ihrer logarithmischen Werte in einem Array ab. Auf diese Weise kann ich die *n_max* Ergebnisse einer Evaluation in einem Graphen mit logarithmischer Skala veranschaulichen.
Für die Marginalisierung der Modellparameter habe ich die Integralgrenzen aus der Anzeige der Messpunkte abgelesen. Dabei habe ich beispielsweise darauf geachtet, die untere Grenze für die Amplitude größer als die y-Messwerte an den gewählten Grenzen der Amplitudenlage anzugeben.
Die Vorgabe für die Anzahl der Stützstellen, die für das Finden des optimalen Rasters verwendet werden sollen, habe ich über den Messpunktabstand in x-Richtung berechnet. So ergibt sich der *vegas*-Eingabeparameter *ncall* aus der Summe der Anzahl von Messpunkten, die gemäß dieser Präzision bzw. dieses Abstandes entlang der einzelnen Dimensions-Achsen des Integralvolumens liegen.
Aufgrund der Schwankungen in den Ergebnissen der VEGAS-Integration, veranlasse ich zehn Evaluationen. Für jede durchlaufene Evaluation wird die Peakanzahl mit dem maximalen Posterior ermittelt. In einem Array, der für jede Peakanzahl (1 bis *n_max*) ein Element besitzt, hält die Anwendung fest, wie oft die einzelnen Peakzahlen die maximale Posterior-Wahrscheinlichkeit innehalten. Am Ende der Anwendung wird das Gesamtergebnis graphisch angezeigt.

3.3.1. Vergleich zur Vorgehensweise von Sivia und Carlile

Sivia und Carlile nehmen für ihre Arbeit an, die Peaks in einem gemessenen Spektrum seien gaußförmig. In dieser Arbeit gehe ich zunächst von Peaks in Form einer Lorentzkurve aus, worauf ich in Abschnitt 3.1 näher eingegangen bin.
Für die Halbwertsbreite der Gauß-Funktionen verwenden Sivia und Carlile einen festen Wert. Diesen schätzen sie und lassen ihren Algorithmus mit unterschiedlichen Schätzwerten für FWHM[4] laufen. Sie geben in ihrem Artikel dazu an, dass sie damit das Programm ihrer Umsetzung als stabiler empfanden. In ihren Berechnungen nehmen sie für die verschiedenen Peaks eines Spektrums die gleiche Halbwertsbreite an.
Ich behandle die Halbwertsbreite als Modellparameter, der für die einzelnen Peaks unterschiedliche, spezifische Werte annehmen kann. Für jeden Peak, der im Spektrum vorhanden sein kann, wird eine Halbwertsbreite marginalisiert. Ich setze für meine Anwendung nicht voraus, dass die Peaks des Spektrums die gleiche Halbwertsbreite besitzen.

Wie bereits im Einleitungskapitel dieser Arbeit erwähnt, löse ich das multiple Integral der Wahrscheinlichkeitsberechnung numerisch. Sivia und Carlile folgen in ihrer Arbeit dem analytischen Lösungsansatz unter Zuhilfenahme der Taylor-Entwicklung. Für ihr Verfahren gehen sie davon aus, dass der größte Beitrag am Integral in der Nähe des Maximums liegt. Dieses umgeben sie durch eine quadratische Taylor-Entwicklung mit einer Parabel. Ich mache keine Annahme darüber, welche Bereiche des Spektrums für das Integral besonders bedeutend sind. In meiner Anwendung sucht der VEGAS-Algorithmus diese Bereiche und gewichtet sie gemäß dem Importance Sampling.

In der Arbeit von Sivia und Carlile gibt die Abszisse die eingegebene Energie ϵ an. Aus diesem Grund können sie den minimalen und maximalen Wert dieses Modellparameters direkt durch die Grenzen der Eingabeenergie der Messung angeben. Bei meinen Messdaten zeigt die Abszisse die chemische Verschiebung δ an. Die Integralgrenzen für die Amplitudenlage lese ich aus der Anzeige der gemessenen Spektrumwerte ab.

[4] Full Width Half Maximum, also Halbwertsbreite

3.3.2. Anwendung simulierter Daten

Um die Anwendung zur Bestimmung der Peakanzahl in einem Spektrum zu testen, habe ich Dummy-Daten erstellt. Ich habe dafür unterschiedlich viele Lorentzkurven zu verschiedenen Spektren aufsummiert.
Die Werte der Modellparameter variierten dabei. Die Messdaten zu den simulierten Spektren entstanden mit den gleichen Modellparameterwerten, aber größeren Abtastschritten in x-Richtung. Zunächst habe ich Spektren aus klar getrennten Peaks angefertigt. Ich habe den Abstand zwischen den Peaks verkleinert, um die Grenzen des Algorithmus zu untersuchen.

Für die erste simulierte Anwendung habe ich eine andere Gleichung der Lorentzfunktion für die Peakform verwendet, als in Abschnitt 3.1 definiert. Die angewandte Funktion ist zu eins normiert und enthält als Modellparameter lediglich die Lage der Peakamplitude μ und die Halbwertsbreite Γ:

$$L^*(x, \mu, \Gamma) = \frac{1}{\pi} \cdot \frac{\Gamma/2}{(x-\mu)^2 + (\Gamma/2)^2}. \tag{3.6}$$

Der Wert der Amplitude ist in dieser Funktionsform durch $A = 2/\pi\Gamma$ berechenbar [35].
Die implementierte Matlab-Anwendung zur Bestimmung der Peakanzahl des simulierten Spektrums mit dieser Peakform ist in Anhang B.3.1 zu sehen.
Da die normierte Lorentzfunktion nicht ideal zur Auswertung von Messdaten ist, habe ich eine gleich gestaltete Anwendung zur Datensimulation und -auswertung mit der Lorentzfunktion aus Gleichung 3.1 implementiert. Der Matlab-Code dazu ist in Anhang B.3.2 aufgeführt.

3.3.3. Umsetzung mit Messdaten

Bei der Anwendung für experimentelle Messdaten habe ich zunächst die Intensitätswerte des Signals über deren Gesamtsumme normiert. Dadurch bleiben die Werte bei der VEGAS-Integration in einem gewissen Rahmen. Die Normierung der Messwerte hat bei den Evaluationen dazu geführt, dass das Integral in einigen Fällen an die Genauigkeitsgrenze (*siehe Abschnitt 3.2.1*) des VEGAS-Algorithmus stieß. Auf diese Weise wurden den Integralen unterschiedlicher Dimensionen fälschlicherweise der gleiche Wert zugeschrieben. Daher habe ich in der *vegas*-Funktion die Einschränkung des kleinstmöglichen Integralwertes für diese Anwendung ausgeschaltet. Ich habe das Integrationsergebnis in der Wahrscheinlichkeitsberechnung mit der Summe skaliert, über die ich zuvor die Messwerte normiert hatte. Dadurch wollte ich bewirken, dass sich die Wahrscheinlichkeitswerte deutlicher unterscheiden.

Um die Dimensionen der Integration zu verringern, habe ich vorerst Schätzwerte für drei Amplitudenlagen, die über ein anderes Verfahren der Bayes'schen Datenanalyse ermittelt wurden, zu Hilfe genommen.
Ich habe die Funktion zur Modellierung der Peakform für die Umsetzung mit den Messdaten abgeändert. Nach einigen Evaluationen der realen Messdaten mit Hilfe der Lorentzfunktion erreichte ich kein eindeutiges Ergebnis für die maximale Posterior-Wahrscheinlichkeit. Ich nahm daraufhin an, dass die reine Lorentzfunktion die Peakform eines gemessenen Spektrums nicht optimal abbilden kann.
Eine Mischform aus einer Gauß- und einer Lorentzfunktion war mein nächster Lösungsansatz.
Dafür habe ich die Faddeeva-Funktion[5] herangezogen. Sie wird zur Beschreibung von elektromagnetischen Reaktionen bei unterschiedlichen physikalischen Problemen verwendet [36].
Aufgespalten in ihren Real- und Imaginärteil, wird sie in der Form: $w(x+iy) = V(x,y) + i \cdot L(x,y)$ geschrieben. Für meine Arbeit nutze ich den Realteil, da dieser das Spektrum bildet (*siehe Abschnitt 2.1.1*). In der Faddeeva-Funktion stellt der Realteil das Voigt-Profil dar. Es beschreibt die Faltung einer Gauß- mit einer Lorentzfunktion. Ich habe für meine Anwendungen eine einfacher berechenbare Gleichung verwendet, die das Voigt-Profil ersetzen kann.
Diese wird als Pseudo-Voigt-Profil (PVP) bezeichnet und stellt eine Linearkombination aus der Gauß-

[5] Die Faddeeva-Funktion war vor 1990 auch als Kramp-Funktion bekannt.

und der Lorentzfunktion dar [32].
Der Anteil der beiden Funktionen ist über eine Variable, η, festgelegt:

$$PVP(x) = \eta \cdot L(x) + (1-\eta) \cdot G(x), \qquad 0 \leqslant \eta \leqslant 1$$
$$\text{mit} \quad L(x) = \frac{A}{1 + \left(\frac{x - x_0}{w/2}\right)^2} \quad , \quad G(x) = A \cdot \exp\left[-ln(2)\left(\frac{x - x_0}{w/2}\right)^2\right] \tag{3.7}$$

$\eta :=$ Anteil der Lorentzfunktion am Pseudo-Voigt-Profil,
$A :=$ Amplitude,
$x_0 :=$ Lage der Amplitude,
$w :=$ Halbwertsbreite (FWHM)

Die drei Modellparameter (A, x_0, w) beider Funktionen stimmen jeweils in ihren Werten überein.
Der Wert für η ist durch eine Kalibrierung bestimmbar. Hierfür kann reines Wasser (H_2O) als Normal eingesetzt werden.
Das Mischverhältnis der Lorentz- und Gaußfunktion, und somit η, ändert sich während eines Messvorgangs nicht. Für den Fall, dass die Messung abgeschlossen ist und die Messdaten ohne kalibrierten η-Wert vorliegen, kann ein Schätzwert verwendet werden.
Im folgenden Abschnitt beschreibe ich einen allgemeinen Ablauf, den ich zur Bestimmung der Peakanzahl eines gemessenen Spektrums implementiert habe.

Bestimmung der Peakanzahl

Zu Beginn lädt meine erstellte Anwendung die gemessenen Daten aus der im Code vermerkten *csv*[6]-Datei und zeigt sie dem Benutzer an.
Wenn das Funktionsmischverhältnis bzw. η bereits durch eine Kalibrierung bestimmt ist, kann der Benutzer den Wert direkt eingeben.
Daraufhin entscheidet der Benutzer, in welchem Bereich der x-Achse er die Daten evaluieren möchte. So können zum Beispiel Multipletts eines Spektrums einzeln untersucht werden (*siehe Abbildung 4.15*). Ich habe meine Auswertung dadurch auf ein gemessenes Multiplett beschränkt.
Nach der Festlegung des Evaluationsbereiches wird der Benutzer zur Eingabe der maximalen Anzahl von Peaks (*n_max*), die sich darin befinden könnten, aufgefordert.
Aufgrund der erwähnten, zuvor geschätzten Amplitudenlagen ist es in meiner Anwendung möglich, beliebig viele solcher Schätzwerte einzugeben. Dabei wird überprüft, ob diese im gewählten Evaluationsbereich liegen.
Weitere wichtige Eingaben durch den Benutzer sind die Minima- und Maxima-Werte für die zu marginalisierenden Amplitudengrößen und Halbwertsbreiten.
Wenn *n_max* die Anzahl eingegebener Schätzwerte der Amplitudenlagen überschreitet, werden auch das Minimum und das Maximum dieses Modellparameters angefordert.
Die Angaben zu den Definitionsbereichen der Modellparameter können in der Anzeige des aktuellen Evaluationsbereiches abgelesen werden.
Nach diesen Eingaben beginnt der Berechnungsablauf des Posteriors der einzelnen Peakzahlen (1 bis *n_max*).
Falls kein Wert für η bekannt ist, ruft meine Anwendung das Curve-Fitting-Tool[7] von Matlab auf.
Ich habe Funktionen implementiert, die Spektren mit einem und bis zu sieben Peaks über das Pseudo-Voigt-Profil darstellen. Diese Funktionen stehen bereit, um durch sie die individuelle Anpassungsgleichung (*Custom Equation*) in das Tool einzugeben.
Des Weiteren legt der Benutzer Anpassungsoptionen fest. Sie beinhalten u. a. Minima- und Maxima-Werte der anzunähernden Funktionsparameter (Modellparameter und η) sowie einen Startwert für

[6] csv: comma separated values
[7] Die Curve-Fitting-Toolbox enthält Funktionen zur Anpassung von Kurven und Oberflächen an gegebene Daten durch Regression [37].

jeden dieser Parameter.
Die Startwerte können experimentell bestimmt werden. Das Fitting-Tool hilft dabei geeignete Werte zu finden, indem es die gemeinsame Anzeige der angepassten Kurve und der Messwerte nach jeder Änderung der Optionen aktualisiert. Um die bereits geschätzten Amplitudenlagen anzugeben, habe ich sie als Startwert des entsprechenden Parameters gesetzt. Für den jeweiligen Minimum- und Maximumwert habe ich einen dazu leicht geringeren bzw. höheren Wert gewählt.
Wenn die Annäherung zufriedenstellend ist, kann aus den Ergebnissen der verwendete η-Wert entnommen werden. Meine Anwendung pausiert ab dem Aufruf des Curve-Fitting-Tools.
Nach der Eingabe des η-Schätzwertes in die Anwendung startet der Ablauf der VEGAS-Integration, wie in Abschnitt 3.2.2 beschrieben.
Die implementierte Matlab-Funktion dieser Anwendung ist in Anhang B.4.1 zu sehen.

Bestimmung der Peakanzahl über numerische Integration mit der Likelihood-Matrix

Da die Verwendung des Pseudo-Voigt-Profils keine Verbesserung der VEGAS-Ergebnisse für die Auswertung real gemessener Daten brachte, habe ich zur Fehlersuche eine zweite Variante der numerischen Integration implementiert.
Die Anwendung dazu folgt dem im vorausgehenden Abschnitt beschriebenen Ablauf. Ihr Code ist in Anhang B.4.2 aufgeführt.
Für die Integration habe ich die Funktion *n_pApw* erstellt, die eine Matrix von Likelihood-Funktionswerten bildet. Um den Integralwert zu berechnen, werden die Werte entlang der einzelnen Matrixdimensionen aufsummiert, wie ich in Abschnitt 3.2 kurz erklärt habe.
Die Likelihood-Matrix wird mit gleichmäßig verteilten Stützstellen generiert. Die Funktion *n_pApw* benötigt daher neben den Integrationsgrenzen den Abstand zwischen den Stützwerten der Modellparameter. Mit diesen Angaben errechnet sie über die quadrierte Abweichung der Funktionswerte für die Stützstellen des modellierten Spektrums von den Messdaten zunächst eine χ^2-Matrix. Diese wird durch einen weiteren Rechenschritt ($\exp[-1/2 \cdot \chi^2]$) zur Likelihood-Matrix.
Ich habe den Abstand der Stützwerte auf den der Messpunkte in x-Richtung gesetzt.
Die Funktion besitzt weitere Eingabeparameter für η, die Messwerte, die Peakanzahl, deren P(N| D,I) bestimmt werden soll, sowie einen Parameter für die Angabe von Schätzwerten der Amplitudenlagen. Dabei kann ein noch offener Wert für η verwendet werden, der dann die Matrix um eine Dimension erweitert.
Ich habe die Bestimmung der Peakanzahl für diese Anwendung auf drei und vier Peaks eingeschränkt.
Die Funktion *n_pApw* gibt die Stützwerte aus, mit denen der χ^2-Wert der Likelihood-Funktion minimal wird.
Parallel dazu habe ich für die Evaluation von P(3 | D, I) einzelne Dimensionen der Likelihood-Matrix nicht mit aufsummiert, um den entsprechenden Modellparameter durch den maximalen Likelihood-Funktionswert zu bestimmen. Die Anwendung zeigt, als Resultat dieser Bestimmung, die Verlaufskurve der Funktionswerte aus der nun zwei-dimensionalen Likelihood-Matrix an.
Bei den ersten Durchläufen der Anwendung erhielt ich linear abfallende Kurven, deren Maximum an dem vorgegebenen, kleinsten Wert des Modellparameters lag. Ich habe diese Werte zur Absicherung mit denen der χ^2-Minimierung verglichen. Das Minimum für den χ^2-Wert sollte mit dem Maximum der Likelihood-Funktion übereinstimmen. Dies war nicht der Fall. Ich habe daher unterschiedliche Änderungen bei der Erstellung der Matrix der möglichen Likelihood-Funktionswerte vorgenommen. Nach einigen Versuchen gelang es mir, eindeutigere Kurven zu berechnen, indem ich den Abstand der Stützwerte für die Amplitude vergrößerte. Zuvor war er so groß gewählt wie der Messwertabstand in x-Richtung. Ich habe für die neue Präzision der Amplitudenstützwerte einen ungefähren Messwertabstand in y-Richtung im Bereich hoher Werte aus der Anzeige des Evaluationsbereiches entnommen. Danach habe ich mir das Spektrum mit den Parameterwerten der χ^2-Minimierung gegenüber dem gemessenen Spektrum im Curve-Fitting-Tool für beide Peakzahlen, drei und vier, angesehen.
Die Modellparameterwerte, die das χ^2 minimieren, habe ich auf die gleiche Weise wie die vorab

geschätzten Amplitudenlagen in das Curve-Fitting-Tool eingegeben.
Die Ergebnisse dieser Anwendung sind in Abschnitt 4.2.2 aufgeführt.
Durch die eben beschriebene Anwendung kam ich zu der Vermutung, dass die unzureichenden Ergebnisse der Anwendung mit dem VEGAS-Algorithmus mit der nicht feststellbaren Präzision der Stützstellen in y-Richtung zusammenhängen könnten.
Daraufhin habe ich die Anwendungen umgeändert, die den VEGAS-Algorithmus zur Bestimmung der Peakanzahl in simulierten und gemessenen Daten verwenden und damit auch über die Amplitude integrieren. Die Änderung bestand darin, die Vorgabe der Stützstellenanzahl mit den unterschiedlichen Präzisionen in x- und y-Richtung zu berechnen. Ich habe, wie für die Anwendung mit der Likelihood-Matrix, einen großen Abstand der Amplitudenstützwerte verwendet.
Die so erzielten Ergebnisse sind in Abschnitt 4.2 dargestellt.
Zur Auswertung der gemessenen Daten mit dem VEGAS-Algorithmus habe ich letztendlich drei verschiedene Wege gewählt. Zunächst verwendete ich, wie in den vorherigen Anwendungen, die drei bereits geschätzten Amplitudenlagen. Da dies nicht den gewünschten Erfolg brachte, habe ich erst über die Modellparameter aller ein bis *n_max* Peaks integriert und schließlich auch über η.

3.4. Ergänzungen zum VEGAS-Code für die Ausgabe des verwendeten Rasters

Um zweidimensionale Teilraster der VEGAS-Integration anzeigen zu können, habe ich nachträglich der *vegas*-Funktion zwei Eingabeparameter hinzugefügt. Zum einen wird die Anzahl an Modellparametern (*nmp*) übergeben. Dies ist vor allem für die Ausgabe der Raster zur Integration für die Bestimmung der Peakanzahl notwendig.
Der zweite Eingabeparameter *rosenb* gibt an, welche Anwendung die Integration verwendet.
Ich habe die Funktion *printGrid* implementiert, die von der *vegas*-Funktion mit u. a. diesen beiden Eingabeparametern aufgerufen wird. Sie bildet die zweidimensionalen Raster zwischen den verschiedenen Dimensions-Achsen ab.
Der Aufruf der Funktion befindet sich in der Hauptiterationsschleife des VEGAS-Algorithmus und ist abhängig von zwei Bedingungen. Über die Angabe des *init*-Wertes und der Dimension *ndim* kann festgelegt bzw. eingeschränkt werden, welche Raster graphisch dargestellt werden sollen.
Im Fall der Integration der Rosenbrock-Funktion erstellt *printGrid* $ndim-1$ Rasterschaubilder. Bei der Rosenbrock-Funktion bilden zwei aufeinanderfolgende Dimensionen die einzelnen Summanden, aus denen sie sich zusammensetzt (*siehe Gleichung 3.5*). Daher lasse ich die Raster zwischen der ersten und der zweiten Dimension, der zweiten und der dritten usw. zeichnen.
Für das Raster der Peakanzahlbestimmung ermittelt die Ausgabefunktion, über die Anzahl der Modellparameter *nmp* und die Dimension *ndim*, die Integraldimensionen der einzelnen Peaks.
So kann sie die Raster zwischen den Dimensions-Achsen der Modellparameter dem entsprechenden Peak zuordnen. Die Abszisse der 2D- Raster gibt dabei die Dimension der Amplitudenlage wieder. Die Ordinate bildet die Dimension der Halbwertsbreite des Peaks oder ggf. der Amplitude ab. In den Abbildungen 4.7, 4.11 und 4.14 im Ergebniskapitel sind Beispiele für zweidimensionale Raster zu sehen, die bei der Bestimmung der Peakanzahl mit *printGrid* gebildet wurden.

4. Ergebnisse

In diesem Kapitel stelle ich Ergebnisse der im vorherigen Kapitel beschriebenen Anwendungen dar. Im ersten Teil werden die Resultate der Beispielanwendung in Form der zwei- und fünf-dimensionalen Rosenbrock-Funktion vorgestellt. Der zweite Teil zeigt die Bestimmung der Peakanzahl in simulierten und real gemessenen Spektren. Alle Berechnungen wurden auf einem 64 Bit Windows 10 Notebook mit einem Inter Celeron N3150 1,6 GHz Prozessor und 8 GB Arbeitsspeicher durchgeführt.

4.1. Berechnung von Integralen der Rosenbrock-Funktion

Ich habe die Rosenbrock-Funktion in zwei und fünf Dimensionen verwendet, um die Integration mit dem von mir implementierten VEGAS-Algorithmus in verschiedenen Dimensionen auszuprobieren. Dazu habe ich den Algorithmus jeweils zehn mal mit den gleichen Eingabewerten laufen lassen.
Das über MuPad errechnete, "wahre" Ergebnis ist in den entsprechenden Ergebnisgraphen durch eine grüne Linie abgebildet. Der Ergebniswert der jeweiligen Integration ist rot markiert. Die dazugehörige Standardabweichung, σ_{best}, ist in Form von Fehlerbalken zu sehen.

4.1.1. Die 2D-Rosenbrock-Funktion

Ich habe das Integral der 2-dimensionale Rosenbrock-Funktion in den Integralgrenzen -5 und 10 über die Variablen beider Dimensionen berechnet. Der Graph der Funktion ist in Abbildung 4.1 zu sehen. Bei den Durchläufen des VEGAS-Aufrufes zur Rasterfindung war die Anzahl an Unterteilungen entlang jeder Dimensions-Achse 44 von den maximal 50 (festgelegt durch die Kontante *NDMX*). Mit dem gefundenen Raster wurden 47 Unterteilungen pro Dimensions-Achse für die genaue Berechnung des Integrals verwendet. Für alle Durchläufe hat der VEGAS-Algorithmus das Importance Sampling und das Stratified Sampling angewandt (*mds* = -1).
In der folgenden Tabelle ist die Anzahl der Stützstellen *ncall* und der Iterationen pro VEGAS-Durchlauf angegeben. Bei den Werten von *ncall* zeigt die grüne Zahl die Anzahl der angewandten Stützstellen an. Die orange Zahl stellt die vorgegebene Anzahl Stützstellen dar.

	ncall	*itmx*
init = 0	100.000	
	96.800	5
init = 1	1.000.000	
	994.050	1

Das Integral ergab mit MuPad 28.692.225.
Zur Veranschaulichung des Integrationsprozesses sind in Tabelle 4.1 die Ergebnisse der vierten VEGAS-Integration exemplarisch aufgeführt. Die Ergebnisse aller zehn VEGAS-Integrationen sind in Abbildung 4.2 zu sehen.
Bei der Berechnung des Integrals entstanden die zwei Raster, die in Abschnitt 2.5 abgebildet sind (Abb. 2.4, Abb. 2.5).

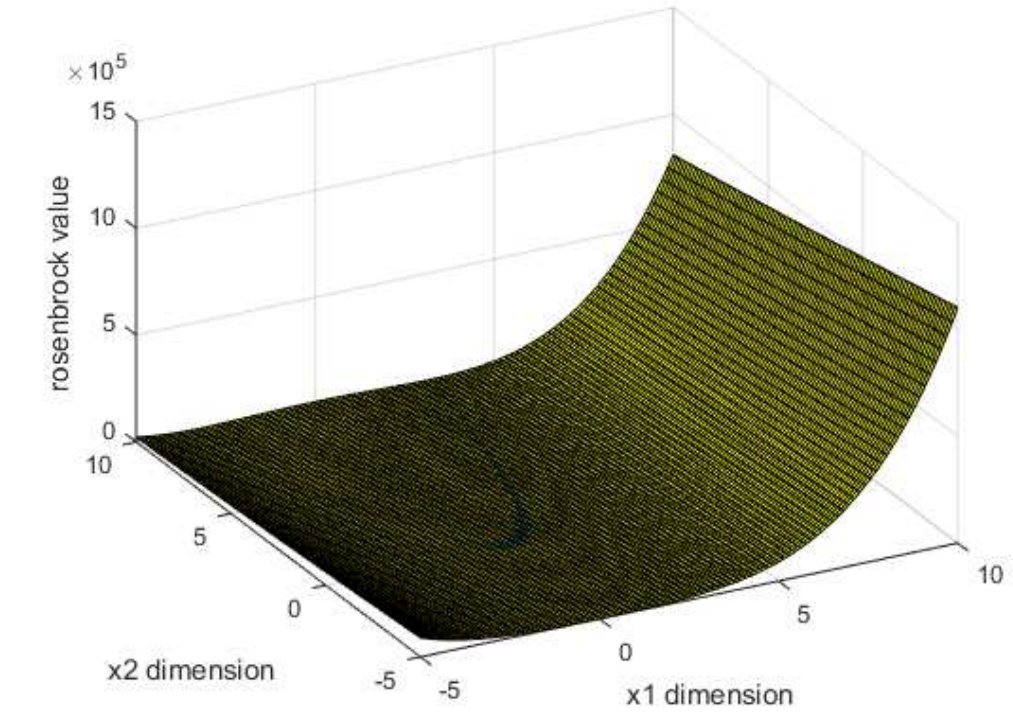


Abbildung 4.1.: 2D-Rosenbrock-Funktion. Die dritte Dimension gibt die Rosenbrock-Funktionswerte an.

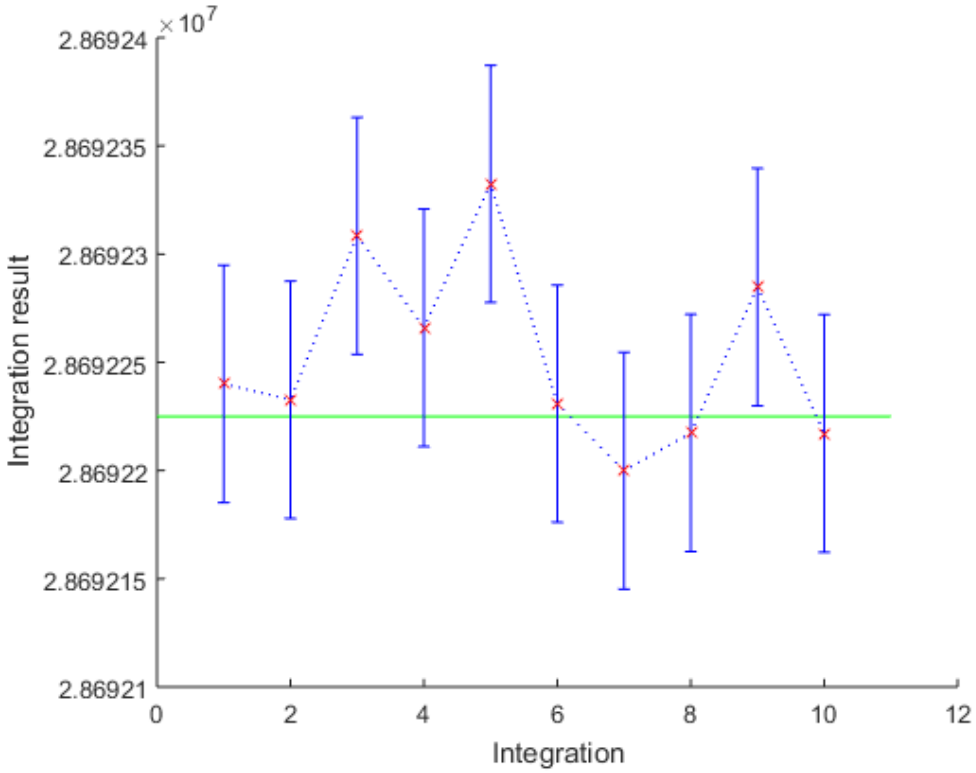


Abbildung 4.2.: Ergebnisse der zehn VEGAS-Integrationen der 2D-Rosenbrock-Funktion mit Fehlerbalken, die die Standardabweichung der Schätzung anzeigen.

		$\hat{I}_i$	σ_i	I_{best}	σ_{best}	χ^2/m
init = 0	1. Iteration	28.689.891,74	1.705,90	28.689.891,74	1.705,90	0
	2. Iteration	28.692.055,36	621,87	28.691.801,57	584,26	1,4
	3. Iteration	28.692.395,61	569,26	28.692.106,31	407,73	0,98
	4. Iteration	28.692.601,35	567,46	28.692.274,86	331,12	0,82
	5. Iteration	28.692.659,33	565,85	28.692.372,93	285,78	0,7
Rechenzeit:	42,629 s					
init = 1	1. Iteration	28.692.265,97	54,86	28.692.265,97	54,86	0
Rechenzeit:	87,59 s (= 1,46 min)					

Tabelle 4.1.: Ergebnisse der 4. VEGAS-Integration der 2D-Rosenbrock-Funktion

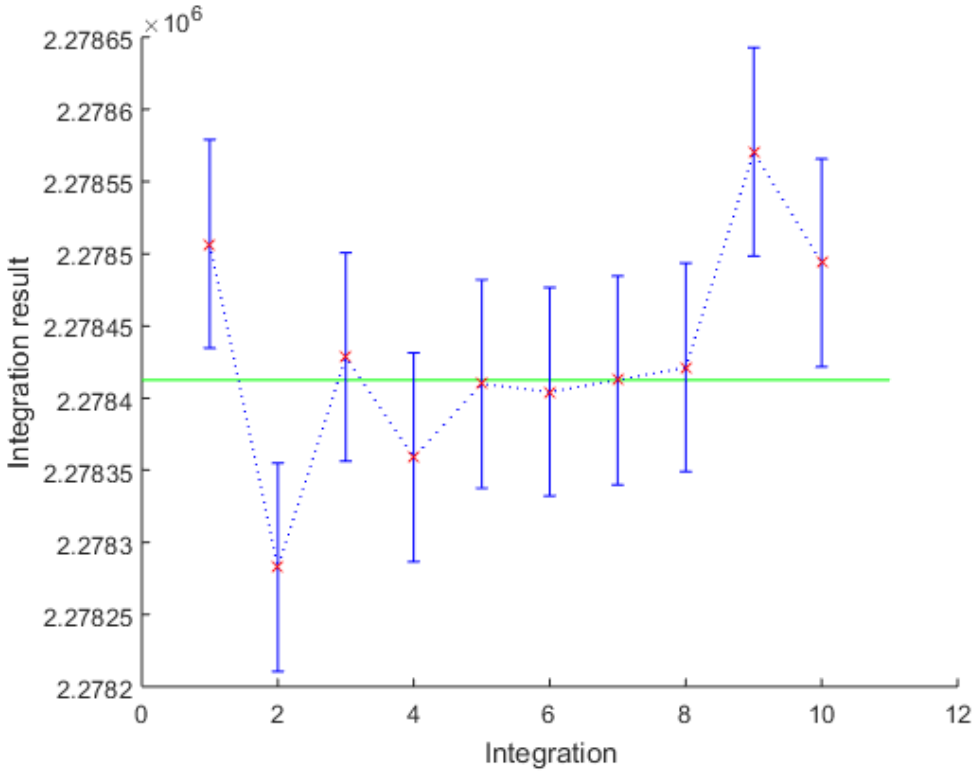


Abbildung 4.3.: Ergebnisse der zehn VEGAS-Integrationen der 5D-Rosenbrock-Funktion mit Fehlerbalken, die die Standardabweichung der Schätzung anzeigen.

4.1.2. Die 5D-Rosenbrock-Funktion

Die Grenzen des 5-dimensionalen Integrals habe ich mit -2,048 und 2,048 für jede Dimension etwas kleiner gewählt. Die Anzahl an Unterteilungen entlang jeder Dimensions-Achse war bei allen Durchläufen gleich, $nd = 50$. Für alle Durchläufe hat der VEGAS-Algorithmus das Importance Sampling verwendet ($mds = 1$).
Das Stratified Sampling wurde aufgrund der Dimensionsgröße durch den Algorithmus ausgeschaltet. Die Angaben zu den Stützstellen, die in der folgenden Tabelle abgebildet sind, zeigen, dass die verwendete Anzahl Stützstellen in grün nicht immer der maximal zugelassenen Stützstellenzahl in orange entspricht.

	ncall	*itmx*
init = 0	300.000	5
init = 1	3.000.000	
	2.839.714	1

Das mit MuPad berechnete, optimale Ergebnis ist 2.278.412,496.
In Abbildung 4.3 sind die Ergebnisse der zehn VEGAS-Integrationen zu sehen. Die einzelnen Berechnungsergebnisse der sechsten VEGAS-Integration zeige ich als Beispiel in
Tabelle 4.2.

Die Abbildungen 4.2 und 4.3 zeigen die in Abschnitt 3.2.2 erwähnten Schwankungen der Ergebniswerte.
Wie in den Tabellen 4.1 und 4.2 exemplarisch zu sehen, ist das Zuverlässigkeitsmaß der Schätzung (χ^2/m) des endgültigen Integrals ($init = 1$) klein bzw. 0. Dies ist bei allen Integrationen der Fall, die nur eine Iteration durchführen.

		$\hat{I}_i$	σ_i	I_{best}	σ_{best}	χ^2/m
init = 0	1. Iteration	2.279.534,55	791,73	2.279.534,55	791,73	0
	2. Iteration	2.279.120,14	398,42	2.279.203,88	355,90	0,22
	3. Iteration	2.278.172,25	348,28	2.278.676,9	248,92	2,3
	4. Iteration	2.278.920,72	343,80	2.278.760,76	201,62	1,6
	5. Iteration	2.278.449,85	344,14	2.278.681,31	173,96	1,4
Rechenzeit:	353,72 s (= 5,9 min)					
init = 1	1. Iteration	2.278.404,26	72,28	2.278.404,26	72,28	0,0012
Rechenzeit:	671,45 s (= 11,19 min)					

Tabelle 4.2.: Ergebnisse der 6. VEGAS-Integration der 5D-Rosenbrock-Funktion

4.2. Berechnung der Peakanzahl in einem Spektrum

In diesem Abschnitt führe ich die Ergebnisse der Anwendungen zur Bestimmung der Peakanzahl in einem Spektrum auf.

4.2.1. Spektrum aus simulierten Daten

Die simulierten Spektren und ihre Messdaten habe ich mit den gleichen Modellparameterwerten erstellt. Dabei ist das Dummy-Spektrum mit einer Präzision von 0,01 zehnmal so genau angegeben wie seine Messdaten.
Für die Auswertungen der simulierten Daten lasse ich die Posterior-Wahrscheinlichkeiten für ein und bis zu sieben Peaks berechnen.
Wegen der Schwankungen in den Ergebniswerten der VEGAS-Integration führen meine Anwendungen zehn Evaluationen mit den gleichen Eingabewerten durch.

Peakform: normierte Lorentzfunktion mit 2 Modellparametern

Ich habe ein Spektrum aus fünf sich teilweise überlagernden Peaks erstellt. Es ist für den δ-Bereich von 0 bis 5 definiert und enthält 50 Messpunkte.
Seine Modellparameter nehmen folgende Werte an:

	δ_0	w
peak 1	1,6	0,13
peak 2	1,84	0,37
peak 3	1,3	0,26
peak 4	3,5	0,35
peak 5	3,15	0,5

Das damit simulierte Spektrum und die erstellten Messdaten sind in Abbildung 4.4 zu sehen.

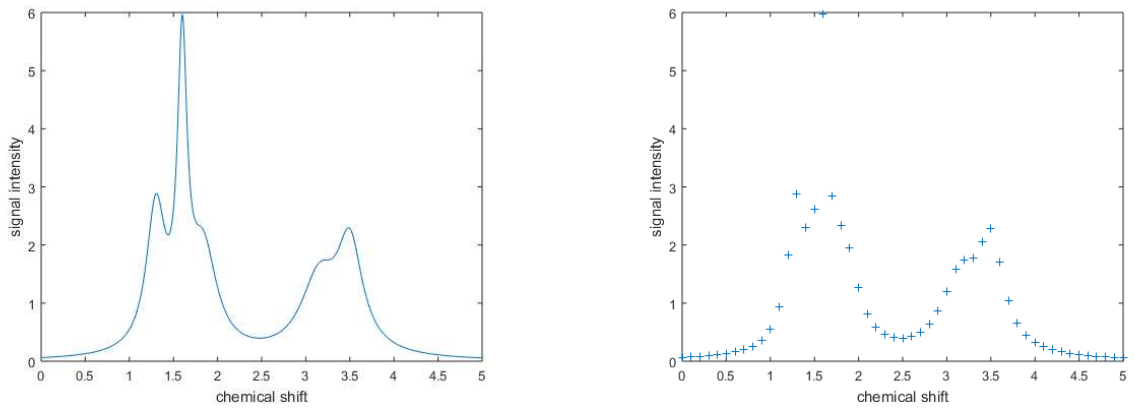


Abbildung 4.4.: Simuliertes Spektrum und zugehörige Messdaten
(5 Peaks, geformt mit der normierten Lorentzfunktion)

Die Integralgrenzen für die Lagen der Amplituden habe ich auf 1 und 4 gesetzt, die Grenzen für die Halbwertsbreite auf 0,1 und 0,8.
Die zehn Evaluationsergebnisse sind in Abbildung 4.5 veranschaulicht. Darin ist zu sehen, dass die Auswertung die wahre Peakanzahl $N = 5$ für das Spektrum bestimmt hat.

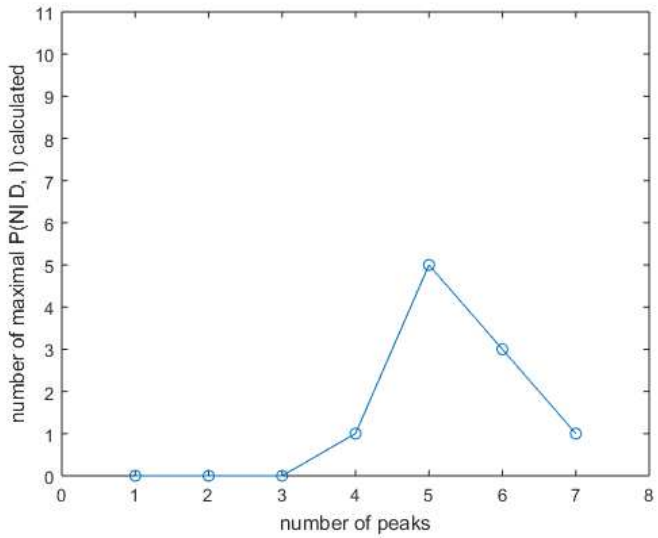


Abbildung 4.5.: Ergebnis der 10 Evaluationen zur maximalen Posterior-Wahrscheinlichkeit der Peakanzahl des simulierten Spektrums, das mit 5 Peaks der normierten Lorentzfunktion erstellt wurde. Jede der zehn Berechnungen ermittelte die Posterior-Wahrscheinlichkeiten für 1 bis 7 Peaks in dem simulierten Spektrum unter Verwendung der VEGAS-Integration.

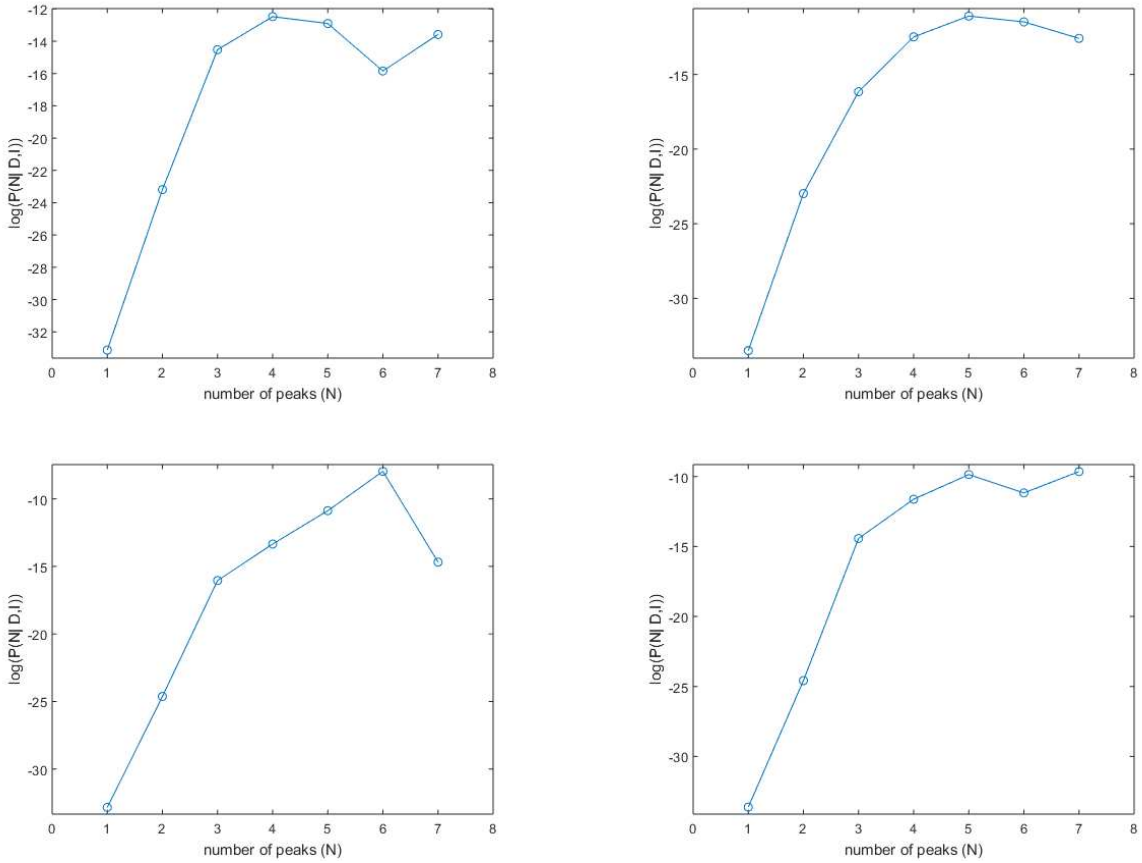


Abbildung 4.6.: Ergebnisbeispiele zu einzelnen Evaluationen der Posterior-Wahrscheinlichkeit der Peakanzahl des simulierten Spektrums, das mit 5 Peaks der normierten Lorentzfunktion erstellt wurde.

In den Graphen der Abbildung 4.6 stelle ich vier Evaluationsergebnisse beispielhaft für die in Abbildung 4.5 sichtbaren Ergebniswerte der wahrscheinlichsten Peakanzahl dar.
Ein Beispiel für das optimale Raster, das bei der VEGAS-Integration erstellt wird, zeigt die Abbildung 4.7 in Form seiner zwei-dimensionalen Teilraster. Die Verwendung dieses Rasters führte zu dem Ergebnis der maximalen Posterior-Wahrscheinlichkeit von 5 Peaks, das in Abbildung 4.6 dargestellt ist.

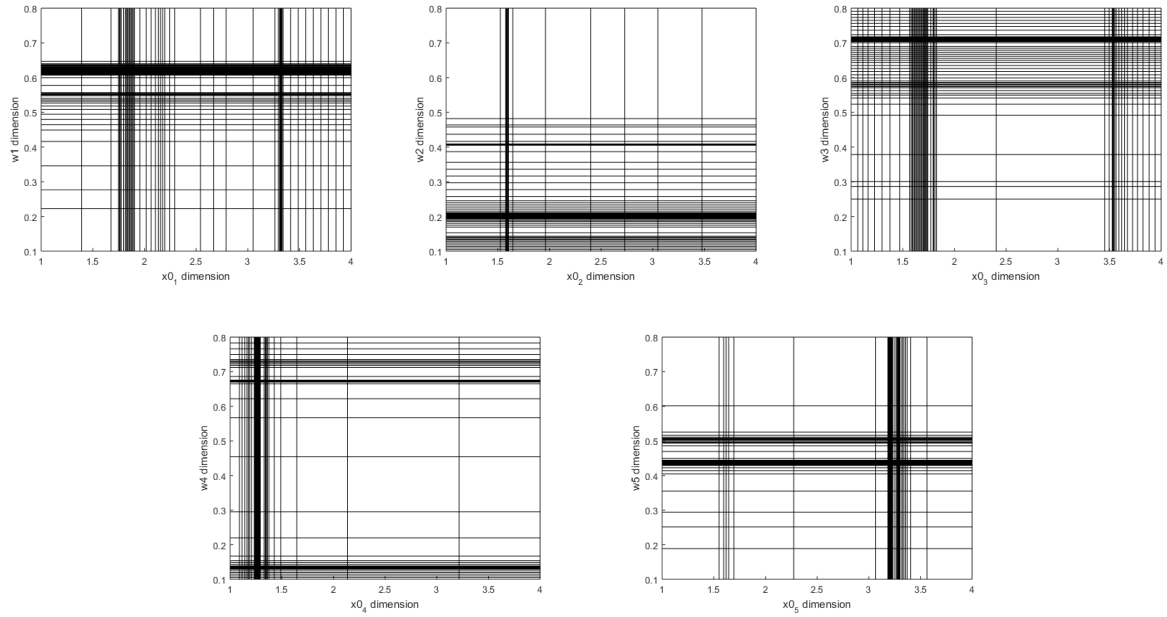


Abbildung 4.7.: Optimales Raster der VEGAS-Integration für eine Evaluation der Posterior-Wahrscheinlichkeit von 5 Peaks.
Es ist dargestellt durch seine zwei-dimensionalen Teilraster. Darin abgebildet sind die einzelnen Integrationsdimensionen der Amplitudenposition δ_0 und der Halbwertsbreite w des jeweiligen Peaks.

Peakform: Lorentzfunktion mit 3 Modellparametern

Für die Simulation eines Spektrums aus Peaks der Lorentzfunktion, die drei unabhängige Modellparameter enthält und in Gleichung 3.1 zu sehen ist, habe ich zwei Peaks erstellt.
Das Spektrum sowie die Messdaten sind entlang der δ-Achse von 0 bis 4 definiert. In diesem Bereich sind 40 Messpunkte gegeben.
Die Werte der Modellparameter sind:

	δ_0	w	A
peak 1	1,3	0,25	2
peak 2	2,5	0,35	2,3

Abbildung 4.8 zeigt das simulierte Spektrum und die erstellten Messdaten.

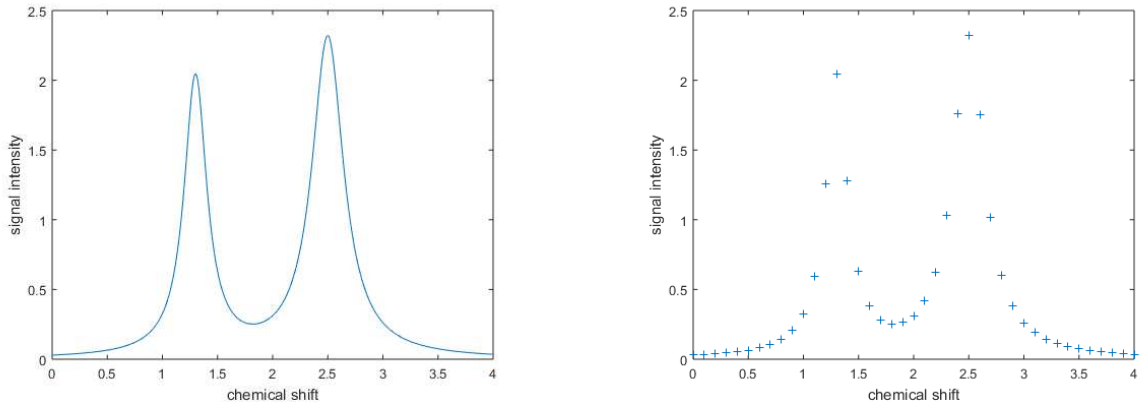


Abbildung 4.8.: Simuliertes Spektrum und zugehörige Messdaten
(2 Peaks, geformt mit der Lorentzfunktion)

Als Integralgrenzen habe ich die folgenden Werte gewählt:

	δ_0	w	A
min	0,5	0,1	0,5
max	3,5	0,5	2,5
Stützwertpräzision	0,1	0,1	0,1

In Abbildung 4.9 sind zehn Evaluationsergebnisse dieser Anwendung dargestellt. Sie zeigen an, dass zwei Peaks für das gegebene Spektrum am wahrscheinlichsten sind.
Für die Ergebniswerte der wahrscheinlichsten Peakanzahl aus dieser Abbildung veranschauliche ich vier Evaluationsergebnisse exemplarisch in den Graphen der Abbildung 4.10.

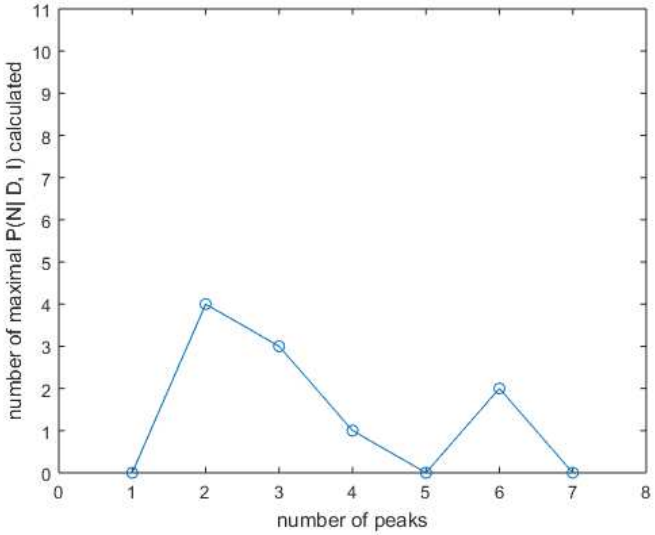


Abbildung 4.9.: Ergebnis der 10 Evaluationen zur maximalen Posterior-Wahrscheinlichkeit der Peakanzahl des Spektrums, das mit 2 Peaks nach der Lorentzfunktion erstellt wurde. Jede der zehn Berechnungen ermittelte die Posterior-Wahrscheinlichkeiten für 1 bis 7 Peaks in dem simulierten Spektrum unter Verwendung der VEGAS-Integration.

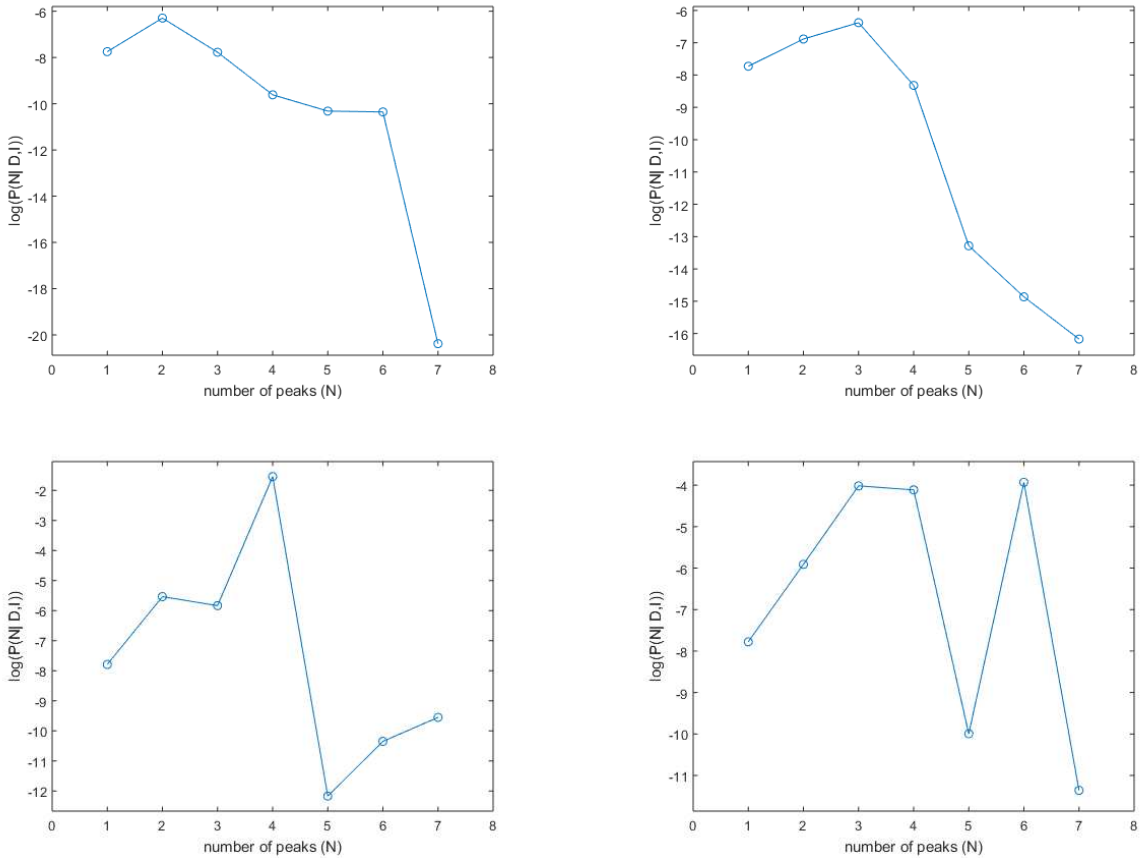


Abbildung 4.10.: Ergebnisbeispiele zu einzelnen Evaluationen der Posterior-Wahrscheinlichkeit der Peakanzahl des Spektrums, das mit 2 Peaks nach der Lorentzfunktion erstellt wurde.

Ein Beispiel des optimalen Rasters für ein Spektrum mit zwei Peaks ist in Abbildung 4.11 zu sehen. Darin sind seine zwei-dimensionalen Teile abgebildet. Das Raster wurde bei der Evaluation gebraucht, die das in Abbildung 4.10 gezeigte Ergebnis der maximalen Posterior-Wahrscheinlichkeit für diese Peakanzahl hervorbrachte.

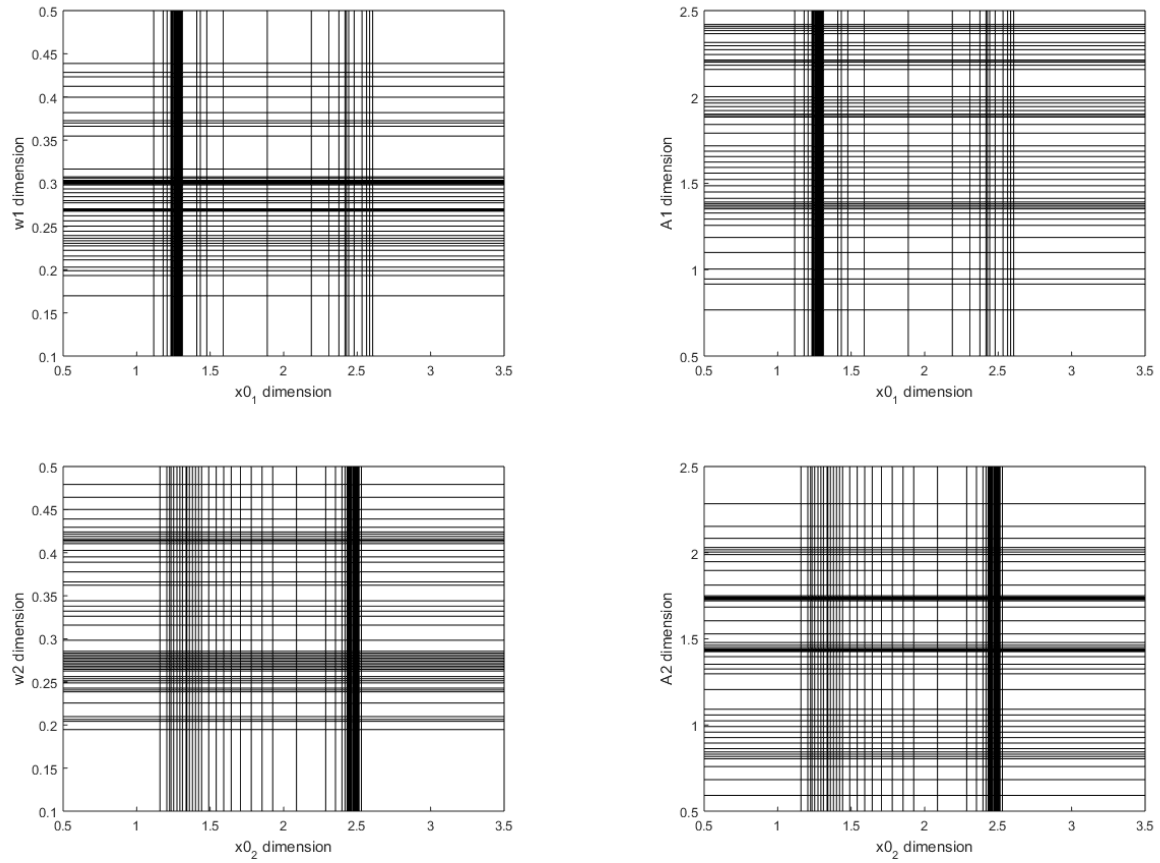


Abbildung 4.11.: Zwei-dimensionale Teile eines optimalen Rasters der VEGAS-Integration für die Berechnung von P(2| D, I). Abgebildet sind die Integrationsdimensionen des 1. und 2. Peaks.

Die angewandte Menge an Stützstellen, die für die jeweilige Peakanzahl in jeder Evaluation gleich ist, sowie die durchschnittlichen Ergebniswerte der zehn Evaluationen sind in Tabelle 4.3 zu sehen. Es ist darin zu erkennen, dass die durchschnittlichen Posterior-Werte für 3, 4 und 6 Peaks größer sind als der Wert der wahren Peakanzahl $N = 2$.

N	*ndim*	*ncall*	$\varnothing I_{best}$	$\varnothing \sigma_{best}$	$\varnothing$ P(N | D,I)
1	3	432	0,001	9,0600 $\cdot 10^{-5}$	4,3211 $\cdot 10^{-4}$
2	6	1.024	0,0067	0,0024	0,0023
3	9	1.536	0,0114	0,0082	0,005
4	12	2.160	0,0331	0,0320	0,0239
5	15	2.700	1,0988 $\cdot 10^{-4}$	8,1634 $\cdot 10^{-5}$	1,6560 $\cdot 10^{-4}$
6	18	3.240	6,8421 $\cdot 10^{-4}$	6,4755 $\cdot 10^{-4}$	0,0026
7	21	3.780	1,5355 $\cdot 10^{-6}$	9,2574 $\cdot 10^{-7}$	1,6873 $\cdot 10^{-5}$

Tabelle 4.3.: Angaben zur VEGAS-Integration und Berechnung von P(N | D,I) des simulierten Spektrums mit drei Modellparametern. Dargestellt sind die Integrationsdimensionen *ndim*, die Anzahl der verwendeten Stützstellen *ncall* für die genaue Integralberechnung sowie die durchschnittlichen Werte der Integrationsergebnisse und der Posterior-Wahrscheinlichkeiten der Peakanzahl der zehn Evaluationen.

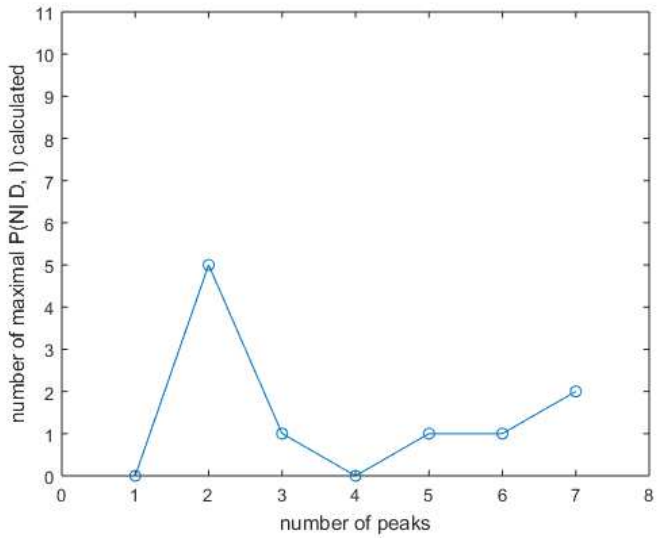


Abbildung 4.12.: Ergebnis der 10 Evaluationen zur maximalen Posterior-Wahrscheinlichkeit der Peakanzahl mit angepasster Präzision in y-Richtung. Jede der zehn Berechnungen ermittelte die Posterior-Wahrscheinlichkeiten für 1 bis 7 Peaks in dem simulierten Spektrum unter Verwendung der VEGAS-Integration.

Peakform: Lorentzfunktion mit 3 Modellparametern - Angepasste Präzision in y-Richtung

Nach der Anwendung der numerischen Integration über die Likelihood-Matrix (*siehe Abschnitt 3.3.3*) habe ich die Präzision der Amplitudenmesswerte für die Berechnung des Eingabeparameters *ncall* angepasst. Statt sie wie zuvor mit der Präzision der Messwerte in x-Richtung gleichzusetzen, habe ich die Stützwertpräzision der Amplituden unter Betrachtung der Messdaten (Abb. 4.8) mit 0,5 angegeben.
Mit ihr habe ich die Vorgabe der Stützstellenanzahl für den VEGAS-Algorithmus berechnet. Die simulierten Messdaten aus dem vorherigen Abschnitt habe ich ansonsten gleichermaßen ausgewertet. Das so erhaltene Ergebnis der durchgeführten zehn Evaluationen ist in Abbildung 4.12 dargestellt. Die Peakanzahl $N = 2$ ist das wahrscheinlichste für das angegebene Spektrum.
Beispiele der Evaluationsergebnisse, die die fünf in Abbildung 4.12 erkennbaren Ergebniswerte der wahrscheinlichsten Peakanzahl wiedergeben, zeige ich in den Graphen der Abbildung 4.13.

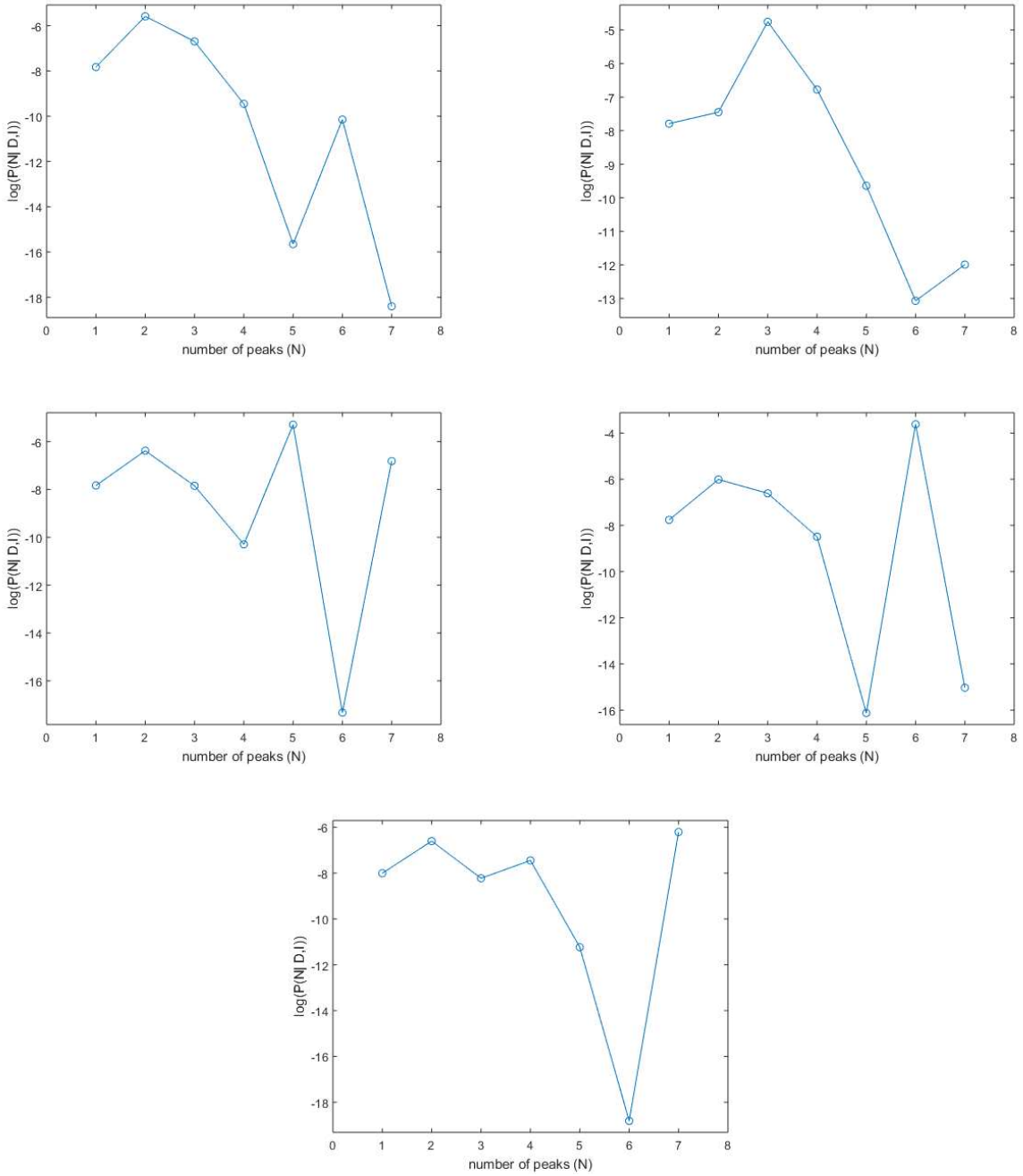


Abbildung 4.13.: Ergebnisbeispiele zu einzelnen Evaluationen der Posterior-Wahrscheinlichkeit der Peakanzahl mit angepasster Präzision in y-Richtung.

Ein Beispiel für das optimale Raster, das bei einer VEGAS-Integration dieser Anwendung verwendet wurde, ist in Abbildung 4.14 veranschaulicht. Es wurde bei der Evaluation erstellt, deren Ergebnis in Abbildung 4.13 die höchste Wahrscheinlichkeit für zwei Peaks angibt.

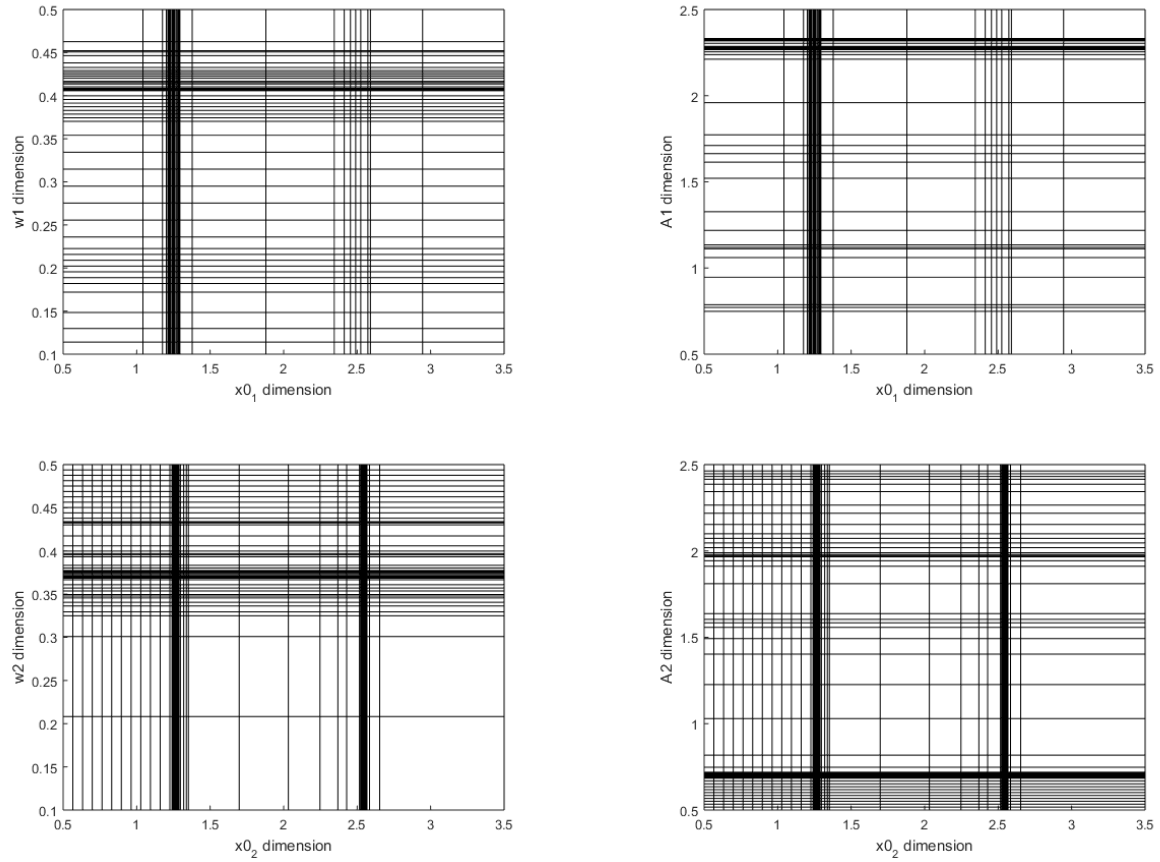


Abbildung 4.14.: Optimales Raster der VEGAS-Integration für die Berechnung von P(2| D, I) mit angepasster Vorgabe der Stützstellenanzahl. Abgebildet sind seine 2D-Teilraster der Integrationsdimensionen des 1. und 2. Peaks.

Die angewandte Menge an Stützstellen, die für die jeweilige Peakanzahl in jeder Evaluation gleich ist, sowie die durchschnittlichen Ergebniswerte der zehn Evaluationen sind in Tabelle 4.4 aufgeführt.

N	*ndim*	*ncall*	$\varnothing I_{best}$	$\varnothing \sigma_{best}$	$\varnothing$ P(N | D,I)
1	3	375	9,4855 $\cdot 10^{-4}$	1,6810 $\cdot 10^{-4}$	3,9523 $\cdot 10^{-4}$
2	6	704	0,0090	0,0051	0,0031
3	9	1.024	0,0052	0,0032	0,0022
4	12	1.520	0,0012	0,0011	8,6190 $\cdot 10^{-4}$
5	15	1.900	3,6588 $\cdot 10^{-4}$	3,4261 $\cdot 10^{-4}$	5,5140 $\cdot 10^{-4}$
6	18	2.280	7,2155 $\cdot 10^{-4}$	7,1756 $\cdot 10^{-4}$	0,0027
7	21	2.660	1,7193 $\cdot 10^{-4}$	1,6710 $\cdot 10^{-4}$	0,0019

Tabelle 4.4.: Angaben zur VEGAS-Integration und Berechnung von P(N | D,I) mit angepasster Vorgabe der Stützstellenanzahl. Dargestellt sind die Integrationsdimensionen *ndim*, die Anzahl der verwendeten Stützstellen *ncall* für die genaue Integralberechnung sowie die durchschnittlichen Werte der Integrationsergebnisse und der Posterior-Wahrscheinlichkeiten der Peakanzahl der zehn Evaluationen.

4.2.2. Spektrum real gemessener Daten

Ich habe die Messdaten, die ich für diese Auswertung verwendet habe, auf einen Evaluationsbereich eingeschränkt. Er umschließt, wie in Abbildung 4.15 zu sehen, entlang der δ-Achse die Messdaten von 136,555 bis 136,59. Dieser Bereich beinhaltet 35 Messpunkte.

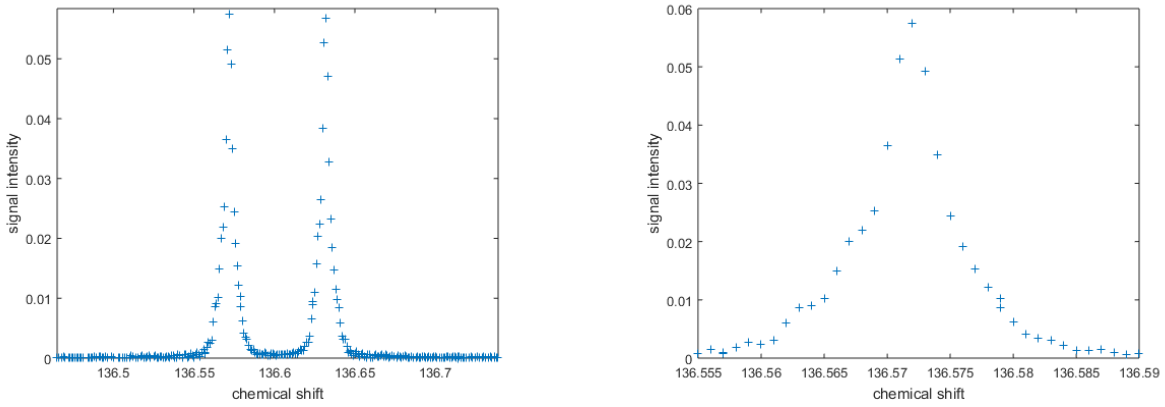


Abbildung 4.15.: Messdaten eines ^{13}C-NMR-Spektrums und eines daraus gewählten Evaluationsbereichs

Wie in Abschnitt 3.3.3 erwähnt, habe ich für die Auswertung der Messdaten des Evaluationsbereiches drei bereits geschätzte Amplitudenlagen hinzugenommen.
Sie wurden zuvor über ein anderes Verfahren der Bayes'schen Datenanalyse ermittelt. Ihre Werte sind: 136,568 $\diamond$ 136,573 $\diamond$ 136,579.
Für die Integration über die Modellparameter habe ich folgende Grenzen festgelegt:

	δ_0	w	A
min	136,56	0,001	0,005
max	136,585	0,013	0,06
Stützwertpräzision	0,001	0,001	0,01

Die kleinste Halbwertsbreite habe ich gleich der δ-Achsenpräzision in den Messdaten gesetzt.
Die maximale Peakanzahl, deren Posterior-Wahrscheinlichkeit berechnet werden soll, ist sieben.
Mit dem anfänglichen Versuch, die Peakanzahl über den maximalen Posterior von Peaks zu bestimmen, die gemäß der Lorentzfunktion verlaufen, erhielt ich das Ergebnis der zehn Evaluationen in Abbildung 4.16.

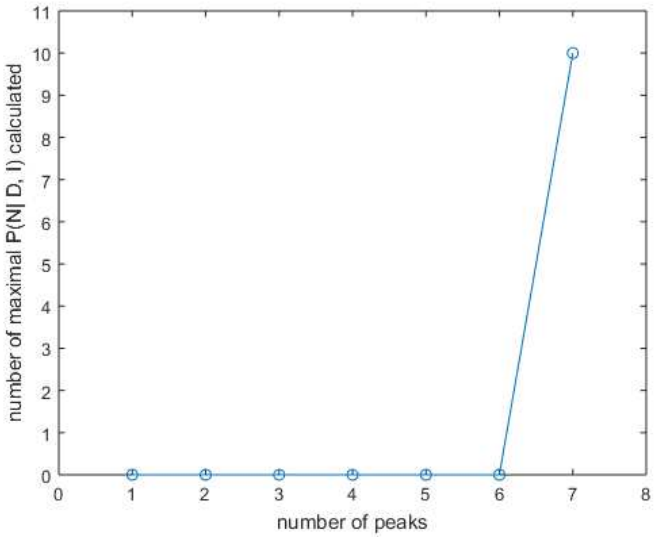


Abbildung 4.16.: Ergebnis der zehn Evaluationen der maximalen Posterior-Wahrscheinlichkeit der Peakanzahl.

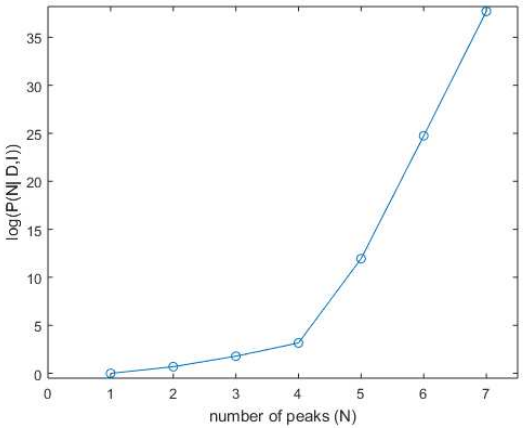


Abbildung 4.17.: Ergebnis einer Evaluation der Posterior-Wahrscheinlichkeit der Peakanzahl.

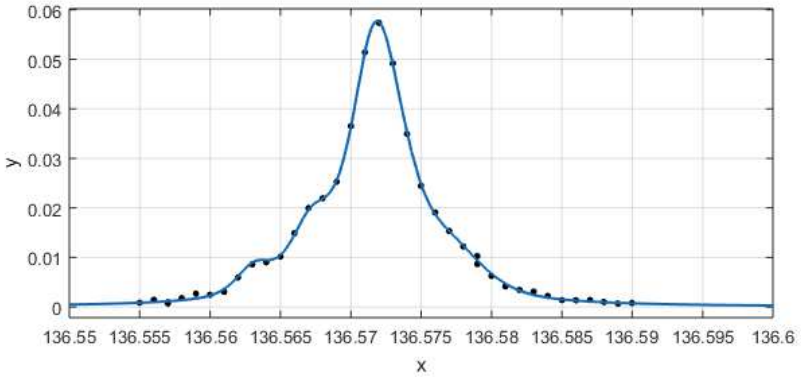


Abbildung 4.18.: Anpassungskurve zur η-Bestimmung für 6 Peaks.

In Abbildung 4.17 ist das Ergebnis einer Evaluation exemplarisch zu sehen. Der Verlauf aller zehn Evaluationsergebnisse war gleich.
Danach habe ich mit der Annahme, die Peaks seien nach dem Pseudo-Voigt-Profil geformt, die weiteren Anwendungen ausgeführt, wie in Abschnitt 3.3.3 beschrieben.
Die Bestimmung des Mischanteils (η) der Lorentzfunktion zur Gaußfunktion mit dem Curve-Fitting-Tool brachte die folgenden Ergebnisse:

n	η
1	1
2	1
3	0,5
4	0,58
5	0,62
6	0,53
7	0,6

Der Graph zur η-Bestimmung für sechs Peaks ist in Abbildung 4.18 als Beispiel veranschaulicht.

Anwendung mit numerischer Integration über die Likelihood-Matrix

Um die Modellparameterbestimmung über den maximalen Likelihood-Funktionswert eindeutiger zu machen, habe ich, wie in Abschnitt 3.3.3 angemerkt, die Abstände der Stützwerte in y-Richtung vergrößert.
Zuvor hatte ich sie auf die Abtastschrittweite der Messwerte in x-Richtung eingestellt, die 0,001 beträgt. Die nun verwendete Präzision der Amplitudenparameter für die Stützstellen ist zehnmal so groß.

Evaluation der Posterior-Wahrscheinlichkeit für drei Peaks

Den verwendeten Mischanteil η im Pseudo-Voigt-Profil habe ich mit den drei geschätzten Amplitudenlagen eines anderen Bayes' schen Verfahrens übernommen. Er beträgt 0,323.
Durch die Marginalisierung der Likelihood-Matrixdimensionen bis auf eine, die den gewünschten Modellparameter repräsentiert, konnte ich die Schaubilder der Abbildung 4.19 herstellen.
Die x-Lage des Maximums eines jeden Schaubildes steht für den Wert des Modellparameters, der ermittelt werden soll. Die Modellparameter des dritten Peaks zeigen in den beiden Graphen der Abbildung kein eindeutiges Maximum an.

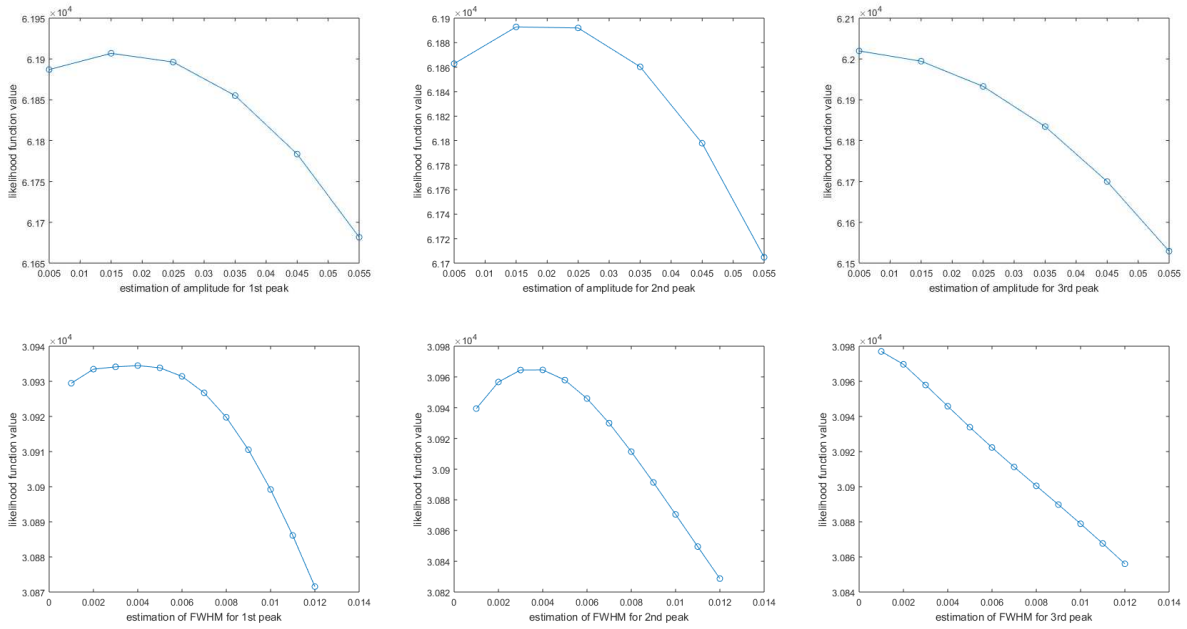


Abbildung 4.19.: Bestimmung der Amplitudenwerte und FWHMs für ein mögliches Spektrum mit 3 Peaks.

Die Werte der Modellparameter, die das χ^2 der Likelihood-Funktion von P(3| D,I) minimierten, sind:

n	A	w
1	0,015	0,008
2	0,045	0,006
3	0,005	0,002

Die Anwendung berechnete die Likelihood-Matrix der sechs Dimensionen in 17,4 Sekunden.
In Abbildung 4.20 ist das Spektrum zu sehen, das mit den χ^2-Minimumwerten der Modellparameter im Curve-Fitting-Tool erstellt wurde. Daneben sind die Abweichungen des Spektrums zu den Messwerten dargestellt.

Die Evaluation mit dieser Anwendung ergab P(3 | D, I) = 7,742939 $\cdot 10^{15}$.

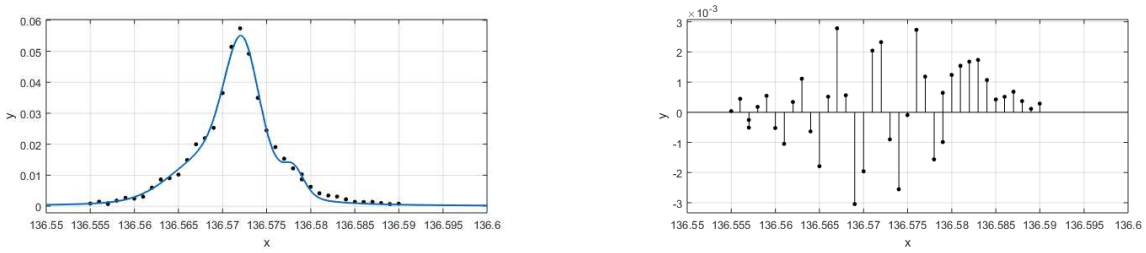


Abbildung 4.20.: Über χ^2-Minimum geschätztes Spektrum im Vergleich zu den Messdaten und Darstellung der Abweichungen.

Evaluation der Posterior-Wahrscheinlichkeit für vier Peaks

Als Erstes habe ich den Wert für η mit dem Curve-Fitting-Tool bestimmt zu 0,336.
Der χ^2-Wert der Likelihood-Funktion von P(4| D,I) wurde mit folgenden Modellparameterwerten minimiert:

n	A	w	δ_0
1	0,015	0,009	
2	0,025	0,007	
3	0,005	0,003	
4	0,025	0,003	136,572

Für die Berechnung der 9-dimensionalen Likelihood-Matrix benötigte die Anwendung 11,02 Stunden. Das mit dem Curve-Fitting-Tool erstellte Spektrum der χ^2-Minimum-Modellparameterwerte ist in Abbildung 4.21 dargestellt. Die Abweichungen zu den Messwerten sind daneben veranschaulicht.

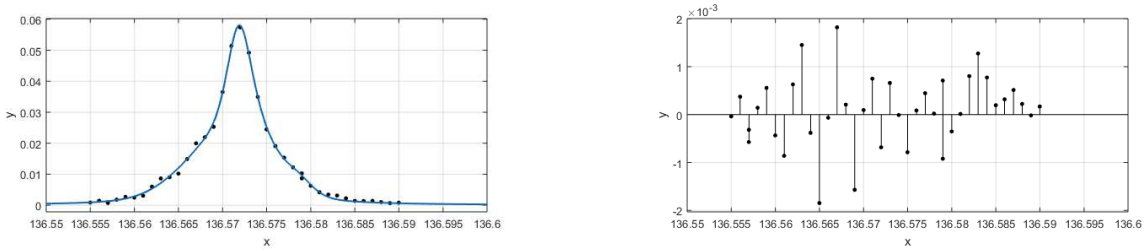


Abbildung 4.21.: Über χ^2-Minimum geschätztes Spektrum im Vergleich zu den Messdaten und Darstellung der Abweichungen.

Die Evaluation ergab P(4 | D, I) = 3,362584 $\cdot 10^{24}$.

Anwendung mit der VEGAS-Integration

Wie am Ende des Abschnitts 3.3.3 beschrieben, habe ich drei Varianten für die Auswertung mit dem VEGAS-Algorithmus umgesetzt.
Die Verwendung der drei Schätzwerte für die Amplitudenlage erzielte Ergebnisse, die beispielhaft durch den ersten Graphen der Abbildung 4.22 dargestellt sind.
Angaben zur beteiligten VEGAS-Integration und Berechnung der Posterior-Wahrscheinlichkeit der Peakanzahl sind in Tabelle 4.5 zu finden.

N	*ndim*	*ncall*	$\varnothing I_{best}$	$\varnothing \sigma_{best}$	$\varnothing$ P(N | D,I)
1	2	162	6,1041 $\cdot 10^{-4}$	1,545 $\cdot 10^{-4}$	0,9253
2	4	324	4,3784 $\cdot 10^{-7}$	1,038 $\cdot 10^{-8}$	2,0103
3	6	512	2,8538 $\cdot 10^{-10}$	4,41 $\cdot 10^{-14}$	5,9558
4	9	950	4,6754 $\cdot 10^{-15}$	1 $\cdot 10^{-16}$	23,6546
5	12	1375	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	153.313,9
6	15	1800	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	5,575051 $\cdot 10^{10}$
7	18	2225	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	2,365173 $\cdot 10^{16}$

Tabelle 4.5.: Angaben zur VEGAS-Integration mit 3 bereits geschätzten Amplitudenlagen und Berechnung von P(N | D,I). Dargestellt sind die Integrationsdimensionen *ndim*, die Anzahl der verwendeten Stützstellen *ncall* für die genaue Integralberechnung sowie die durchschnittlichen Werte der Integrationsergebnisse und der Posterior-Wahrscheinlichkeiten der Peakanzahl der zehn Evaluationen.

Mit dem zweiten Auswertungsansatz habe ich über die Modellparameter aller Peaks integriert. Die Werte der Integration und die resultierenden Posterior-Wahrscheinlichkeiten sind in Tabelle 4.6 aufgelistet. Im zweiten Graphen der Abbildung 4.22 ist das Beispiel eines Evaluationsergebnisses zu sehen.

N	*ndim*	*ncall*	$\varnothing I_{best}$	$\varnothing \sigma_{best}$	$\varnothing$ P(N | D,I)
1	3	375	1,6816 $\cdot 10^{-5}$	1,101 $\cdot 10^{-6}$	1,0192
2	6	832	2,7025 $\cdot 10^{-10}$	3,54 $\cdot 10^{-14}$	1,9853
3	9	1024	4,4415 $\cdot 10^{-15}$	1 $\cdot 10^{-16}$	5,9324
4	12	1700	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	32.379,9
5	15	2125	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	9,81209 $\cdot 10^{9}$
6	18	2550	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	3,568033 $\cdot 10^{15}$
7	21	2975	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	1,513711 $\cdot 10^{21}$

Tabelle 4.6.: Angaben zur VEGAS-Integration über die Modellparameter aller Peaks und Berechnung von P(N | D,I). Dargestellt sind die Integrationsdimensionen *ndim*, die Anzahl der verwendeten Stützstellen *ncall* für die genaue Integralberechnung sowie die durchschnittlichen Werte der Integrationsergebnisse und der Posterior-Wahrscheinlichkeiten der Peakanzahl der zehn Evaluationen.

Bei der letzten Auswertungsvariante habe ich nicht nur die Modellparameter, sondern auch den Mischanteil η des Pseudo-Voigt-Profils marginalisiert. Dies ergab gleiche Wahrscheinlichkeitswerte wie ohne Integration über η. Daher ist das exemplarische Evaluationsergebnis ebenfalls durch den zweiten Graphen der Abbildung 4.22 veranschaulicht.
In Tabelle 4.7 sind die Integrationswerte und berechneten Posterior-Wahrscheinlichkeiten der Peakanzahl aufgeführt. Bei der Evaluation mit den drei gegebenen Amplitudenlagen und der, die über die Modellparameter aller Peaks integriert, benötigten die Anwendungen mehrere *vegas*-Aufrufe, um

N	*ndim*	*ncall*	$\varnothing I_{best}$	$\varnothing \sigma_{best}$	$\varnothing$ P(N | D,I)
1	4	1250	1,6419 $\cdot 10^{-5}$	1,44 $\cdot 10^{-7}$	0,9928
2	7	1792	2,702 $\cdot 10^{-10}$	2,6 $\cdot 10^{-14}$	1,9849
3	10	2048	4,4418 $\cdot 10^{-15}$	1 $\cdot 10^{-16}$	5,9324
4	13	2700	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	32.379,9
5	16	3125	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	9,81209 $\cdot 10^{9}$
6	19	3550	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	3,568033 $\cdot 10^{15}$
7	22	3975	1 $\cdot 10^{-16}$	1 $\cdot 10^{-16}$	1,513711 $\cdot 10^{21}$

Tabelle 4.7.: Angaben zur VEGAS-Integration über die Modellparameter aller Peaks und η sowie zur Berechnung von P(N | D,I). Dargestellt sind die Integrationsdimensionen *ndim*, die Anzahl der verwendeten Stützstellen *ncall* sowie die durchschnittlichen Werte der Integrationsergebnisse und der Posterior-Wahrscheinlichkeiten der Peakanzahl der zehn Evaluationen.

ein zuverlässiges Raster für die Integration über die zwei bzw. die drei Dimensionen eines Peaks zu finden.
Die durchschnittliche Zeit, in der der VEGAS-Algorithmus das Integral für sieben Peaks über 22 Dimensionen berechnete, beträgt 4,2 Minuten.

Wie im ersten Graphen der Abbildung 4.22 zu sehen, steigen die logarithmischen Posterior-Werte für die Anwendung mit drei angegebenen Amplitudenlagen zunächst für ein bis zu vier Peaks gleichmäßig an und verlaufen dann steiler bis zum Maximum bei sieben Peaks. Der zweite Graph zeigt das Ergebnis der zwei Auswertungsvarianten ohne zuvor geschätzte Werte an. Das Maximum liegt auch hier bei sieben Peaks. Die Änderung der Steigung ist bereits ab der Wahrscheinlichkeit für drei Peaks zu erkennen.

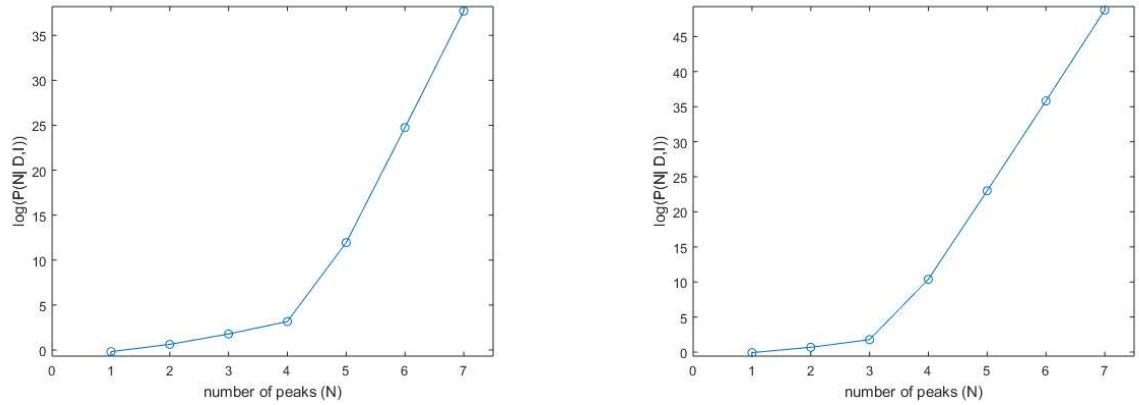


Abbildung 4.22.: Exemplarische Ergebnisse der drei durchgeführten Auswertungsvarianten.
1) Evaluationsergebnis, für das drei bereits geschätzte δ_0-Werte verwendet wurden.
2) Ergebnis einer Evaluation ohne zuvor geschätzte Amplitudenlagen. Der Graph zeigt das Ergebnis der Integration über die Modellparameter aller Peaks und der entsprechenden Integration mit zusätzlicher Dimension für η, da beide das gleiche Ergebnis ergaben.

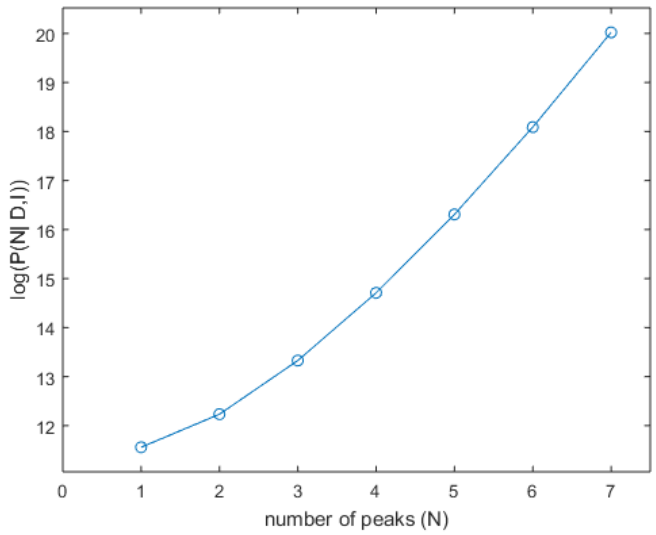


Abbildung 4.23.: Evaluationsergebnis, das durch die VEGAS-Integration über alle Modellparameter der 1 bis 7 Peaks entstand.

Mit der Aufhebung des kleinstmöglichen Integrationsergebnisses, die in Abschnitt 3.3.3 erläutert ist, nehmen die einzelnen Evaluationsergebnisse eine durchgehend lineare Verlaufsform an. Abbildung 4.23 zeigt einen Graphen, der das Evaluationsergebnis aller drei Auswertungsvarianten beispielhaft wiedergibt. Die Endergebnisse der jeweils zehn Evaluationen der drei Anwendungen stimmen mit dem Ergebnisgraphen in Abbildung 4.16 überein. Demnach sind 7 Peaks für das gegebene Spektrum am wahrscheinlichsten.

5. Diskussion

In diesem Kapitel evaluiere ich die erstellten Anwendungen, meine verwendeten Methoden und die Ergebnisse.

5.1. Bewertung der Anwendung des VEGAS-Algorithmus

Die Bedeutung einiger Variablen im Original-VEGAS-Algorithmus ist nicht klar zu erfassen. Dies erschwerte die Einarbeitung und erhöhte den Zeitaufwand über das angedachte Maß hinaus. Im Verlauf des Algorithmus werden die Variablen durch Zuweisung neuer Werte auch in ihrer Bedeutung abgeändert. Ein einfaches Beispiel hierfür ist die Variable *ncall*. Sie wird der *vegas*-Funktion als gewünschte Anzahl der Stützstellen für die Integration übergeben (*siehe Abschnitt 2.5*). Mit dieser vorgegebenen Stützstellenanzahl wird die Menge der Stützstellen berechnet, die tatsächlich erstellt und für die Integration verwendet wird. Die Variable *ncall* wird mit diesem errechneten Wert neu belegt. In den Ergebnissen der Beispielanwendung habe ich die zwei *ncall*-Werte der Integration verschiedenfarbig angegeben (*siehe Abschnitte 4.1.1 und 4.1.2*). Die Werte anderer *vegas*-Variablen und ihre Bedeutung werden ähnlich oder noch drastischer verändert.
Neue Variablen bei jeder Bedeutungsänderung einzuführen, ist eine Möglichkeit, um dieses Durcheinander zu beheben. Im Rahmen der Diplomarbeit wäre dies jedoch zu komplex geworden.

Dadurch, dass meine Anwendungen nur eine Iteration zur genauen Ermittlung des Integrals durchführen, ist deren Zuverlässigkeitsmaß χ^2/m immer sehr klein bzw. 0. Ich überprüfe auch deshalb die berechnete Verlässlichkeit für das gefundene Raster, um zu gewährleisten, dass die Schätzung des Integrals zuverlässig ist.
Da der VEGAS-Algorithmus aufgrund des darin umgesetzten Monte-Carlo-Verfahrens mit Zufallszahlen arbeitet, kommt es zu Schwankungen in den Ergebnissen.
Um das "wahre" Ergebnis festzustellen, sind einige VEGAS-Integrationen mit den gleichen Eingabewerten notwendig. Wie viele Integrationen durchgeführt werden müssen, um die Tendenz zum Ergebniswert erkennen zu können, ist experimentell herauszufinden. Für meine Anwendungen habe ich zehn Evaluationen verwendet. Dadurch konnte ich die Ergebnisse plausibel bestimmen.
Die Stützstellen werden innerhalb des Integrationsbereiches zufällig erzeugt. Da sie gaußverteilt um die wahren Funktionswerte liegen, deckt ihre geschätzte Standardabweichung nicht immer den wahren Wert ab. Gemäß der Gaußverteilung schließen somit etwa 68 % der Zufalls-Stützstellen mit ihrer Standardabweichung den wahren Wert ein, 32 % erreichen ihn jedoch nicht. Dies führt dazu, dass die berechneten Integrale der einzelnen Evaluationen und ihre Standardabweichungen den wahren Integralwert zu 32 % nicht erfassen ($\sigma_{best} < |I_{best} - I|$). In den Abbildungen 4.2 und 4.3 ist dies deutlich dargestellt.

5.1.1. Entscheidung für Matlab gegenüber Python

Ich habe die Anwendungen des VEGAS-Algorithmus auch mit dem Python-Programm von G. P. Lepage ausgeführt. Damit habe ich meinen Algorithmus, der auf dem ursprünglichen Fortran- bzw. C-Code basiert, im Vergleich zur neuen VEGAS-Version in Python (*siehe Abschnitt 2.5.1*) erprobt.
Ich habe die Integrand-Funktion in Python codiert und die Parameter, soweit möglich, entsprechend meiner bisher verwendeten Werte gesetzt. Die Anzahl der anzuwendenden Stützstellen, der Rasterunterteilungen und der Algorithmus-Iterationen habe ich aus meinen Anwendungen übernommen. Mit vergleichbaren Parameterwerten und einer Batch-Größe von 1 fielen die Ergebnisse der neuen

VEGAS-Version nicht genauer aus als die meiner bisherigen Matlab-Version.
Ich habe mich außerdem dafür entschieden, die Implementierung in Matlab fortzusetzen, da ich zur Schätzung des Mischanteils η und zur Veranschaulichung der Übereinstimmung mit den Messdaten das Curve-Fitting-Tool von Matlab verwendet habe.
Ein weiterer Grund für die Entscheidung war die Möglichkeit, dass die Implementierung zukünftig wissenschaftlich weiterverwendet wird.
In Matlab kann die Einarbeitung und mögliche Weiterentwicklung auch ohne große programmiertechnische Kenntnisse und damit einfacher als in Python gelingen.

5.1.2. Auswertung der Integration der Rosenbrock-Funktion

Da sie eine stetige und quadratisch integrierbare Funktion darstellt, eignet sich die Rosenbrock-Funktion zur Untersuchung des VEGAS-Algorithmus. Sie ist beliebig genau abtastbar, so dass ich die Anzahl der Stützstellen frei wählen konnte. Ich habe das Integral lediglich für die zwei- und fünfdimensionale Rosenbrock-Funktion berechnet, da die Berechnung des "wahren" Integralwertes mit MuPad für höhere Dimensionen unangemessen lange dauert.
Um die Handhabung der VEGAS-Integration zu erproben, waren die zwei und fünf Dimensionen ausreichend. Die erzielten Integrationsergebnisse unterstützten die Wahl des VEGAS-Algorithmus zur Lösung des multiplen Integrals der Wahrscheinlichkeitsberechnung.

5.2. Erörterung der umgesetzten Bestimmung der Peakanzahl

In meiner Anwendung werden die Eingaben der Integrationsgrenzen nicht kontrolliert. Der Benutzer legt daher beispielsweise alleine fest, ab welchem Wert der Signalintensität er diese als Amplitude ansieht. Wenn in den Messdaten offensichtlich Rauschen vorhanden ist, sollte der Benutzer dieses so gut wie möglich von der Auswertung ausschließen, indem er die Amplitudenwerte entsprechend nach unten begrenzt.
Wie in Abschnitt 3.3 beschrieben, berechnen meine Anwendungen die Posterior-Wahrscheinlichkeiten der Peakzahlen mehrmals mit den gleichen Eingabewerten. Der Grund dafür sind die Schwankungen in den Ergebnissen der VEGAS-Integration (*siehe Abschnitt 3.2.2*).
Es ist auch möglich, zuerst den "wahren" Wert des Integrals für eine Peakanzahl durch mehrere Integrationen zu ermitteln und mit diesem dann ein Mal den dazugehörigen Posterior P(N| D, I) zu berechnen. Da die Integration der aufwendigste Teil der Wahrscheinlichkeitsberechnung ist, habe ich entschieden, die gesamte Bestimmung der Peakanzahl mehrere Male laufenzulassen. Die Tendenz zum "wahren" Integralwert bzw. zur damit errechneten maximalen Posterior-Wahrscheinlichkeit kann so im letzten Graphen, den die Anwendung ausgibt, erkannt werden. Der Graph zeigt die Häufigkeit an, mit der die einzelnen Peakzahlen die maximale Posterior-Wahrscheinlichkeit in den zehn Evaluationen hatten. Die Schwankungen der Wahrscheinlichkeiten sind in den Ergebnisgraphen der einzelnen Evaluationen zu sehen.
Ich denke meine Umsetzung ist einfacher und zielführender, da ich die Ergebnisse in direktem Bezug zu den Messdaten am Ende der Anwendung auswerten kann.
Bei der alternativen Möglichkeit würden die Integrationsergebnisse während der Anwendung ausgewertet. Dabei können die Auswirkungen eines bestimmten Integralwertes auf die Posterior-Wahrscheinlichkeit noch nicht sicher abgeschätzt werden.

5.2.1. Vergleich zum Vorgehen von Sivia und Carlile

Sivia und Carlile verwenden zur analytischen Berechnung des multiplen Integrals, wie in den Abschnitten 2.3 und 3.3.1 beschrieben, die Taylorentwicklung zweiter Ordnung. Mit der Parabel, die sie um den Maximalwert der Integrand-Funktion legen, schließen sie die kleineren Ausläufe der Funktionskurve aus ihren Betrachtungen aus. Diese Nebenbereiche des Integranden entfallen und werden nicht in den weiteren Berechnungen bedacht. Dies kann von Nachteil sein, da die Werte aus diesen

Bereichen nicht absolut bedeutungslos für das Ergebnis sein müssen. In meinem Verfahren gehen alle Bereiche des Integrals in dessen Berechnung ein. Sie werden im Laufe der Integration lediglich durch das Importance Sampling unterschiedlich gewichtet.
Sowohl Sivia und Carlile als auch ich haben für die Bestimmung der Peakanzahl χ^2 verwendet. Dies basiert auf der Annahme, dass die Messpunkte gaußverteilt um die wahren Signalwerte liegen. Um die Berechnung von P(N| D, I) mit χ^2 einwandfrei rechtfertigen zu können, ist ein experimenteller Nachweis erforderlich. Dieser steht noch aus.
Wie Sivia und Carlile habe ich einen gleichmäßigen Prior für die Modellparameter angenommen (*siehe Abschnitt 3.1*). Dies vereinfacht die Berechnung der Posterior-Wahrscheinlichkeiten. Ebenso gehe ich davon aus, dass die Form des Priors die Auswertung nur schwach beeinflusst. Dies ist der Fall, wenn die zwei oder drei Parameter je Peak aus verhältnismäßig vielen gemessenen Daten geschätzt werden (*siehe Abschnitt 2.3*).
Bei den durchgeführten Auswertungen ist das Verhältnis zwischen der Anzahl marginalisierter Parameter und der Anzahl an Messwerten bezüglich dieser Voraussetzung nicht ideal.
Für die Marginalisierung von drei Modellparametern pro Peak, die bis zu 21 Parameter umfasst, stehen in den simulierten Daten 40 und im Evaluationsbereich der realen Daten 35 Messwerte zur Verfügung. Die Auswertung der simulierten Daten erfüllt die Voraussetzung vermutlich in ausreichendem Maße. Mit einer Anzahl von Messwerten, die im Vergleich zu den maximal 21 Modellparametern fast doppelt so groß ist, konnte die Bestimmung der Peakanzahl im simulierten Spektrum erfolgreich durchgeführt werden. In den Auswertungen der real gemessenen Daten ist der Mengenunterschied anscheinend nicht groß genug, was u. a. zu den mangelhaften Ergebnissen führte. Es kann versucht werden, die Auswertung durch einen abgeänderten Prior der Modellparameter zu verbessern.
Ein Prior, der beispielsweise durch analytische Überlegungen hinsichtlich der Werteverteilung der Modellparameter gestützt wird, macht die Auswertung unabhängiger von dem zuvor genannten Mengenverhältnis.
In meiner Arbeit habe ich festgelegt, dass die Standardabweichung σ_k der Messwerte konstant sei. Dies ist bisher nicht belegt. Die Annahme erlaubte im Rahmen dieser Diplomarbeit eine einfachere Durchführung der Wahrscheinlichkeitsberechnung. Ein experimenteller Nachweis wäre dennoch an dieser Stelle wünschenswert.

5.2.2. Auswertung simulierter Daten

Die Lorentzfunktion mit zwei Modellparametern (*siehe Gleichung 3.6*), die ich für die erste simulierte Anwendung verwendet habe, ist normiert zu eins. Aus diesem Grund war es von Anfang an nicht sinnvoll, sie zur Bestimmung der Peakanzahl in einem gemessenen Spektrum zu verwenden.
Auch die durch die Funktion gegebene Proportionalität der Amplitude zur Halbwertsbreite der Lorentzkurve ist, aus meinem Verständnis der NMR-Spektroskopie, nicht für die Anwendung geeignet. Wie in Abschnitt 2.1.1 angeführt, kann über die Größe der Amplitude auf die Anzahl der Atome in einer Gruppe geschlossen werden. Das Integral unter einem Peak, das von der Amplitude und der Halbwertsbreite abhängig ist, gibt die Anzahl gleichwertiger Atomkerne in einer Verbindung an. Daher ist es nicht plausibel, für das Peakmodell eine Funktion zu wählen, in der sich die Amplitude und die Halbwertsbreite beeinflussen.
Dennoch war die Verwendung dieser Form der Lorentzfunktion hilfreich, um die Bestimmung der Peakanzahl mit dem VEGAS-Algorithmus erfolgreich durchzuführen und im Vergleich zur Anwendung der Peakform mit drei Modellparametern zu betrachten.
An der Verlaufskurve der logarithmischen Posterior-Werte einer Evaluation kann der Benutzer das wahre Ergebnis der Peakanzahl tendenziell erkennen. Wenn die Werte für mehrere aufeinanderfolgende Peakzahlen sehr schwanken, deutet dies auf Ausreißer durch die Verwendung der Zufallszahlen hin. In Abbildung 4.13 sind solche Sprünge der Ergebniswerte in den drei letzten Graphen zu sehen. Ausreißer, die sogar die maximale Wahrscheinlichkeit erreichen, kommen durch eine nicht ausreichende Anzahl von Stützstellen zustande. Die genannte Abbildung zeigt Ergebnisse der Integration, für die ich die Vorgabe der Stützstellenanzahl mit der angepassten Amplitudenpräzision errechnet habe.

Da die geänderte Präzision bzw. der Abstand der Amplitudenstützwerte größer als der vorherige ist, werden weniger Stützstellen für diese Integration verwendet. Daher gibt es in den Integrationsergebnissen ohne Präzisionsanpassung in y-Richtung nicht so viele Ausreißer (*siehe Abbildung 4.10*). Das Endergebnis, das die Ergebnisse aller Evaluationen darstellt, liefert eine quantitative Bestätigung dafür, dass das Maximum einer gleichmäßig verlaufenden Kurve von logarithmischen Wahrscheinlichkeiten dem endgültigen Maximum am ehesten entspricht.
Dies ist beispielsweise in Abbildung 4.6 zu beobachten. Der Graph, dessen Wahrscheinlichkeitswerte den gleichmäßigsten Verlauf haben, gibt, wie das Endergebnis in Abbildung 4.5, die wahrscheinlichste Peakanzahl $N = 5$ an.

Die Anzeige der optimalen 2D-Raster der Integrationen mit den simulierten Daten ermöglicht eine Abschätzung der Modellparameterwerte für das Spektrum. Ungeachtet der Tatsache, dass das optimale Raster nicht zu diesem Zweck erstellt wird, gibt es Aufschluss über die möglichen Modellparameterwerte des Spektrums.
Besonders eindeutig in den Rastern zu bestimmen, sind die Werte der simulierten Amplitudenlagen, wie zum Beispiel in Abbildung 4.14.
Die Halbwertsbreiten, die für die Datensimulation verwendet wurden, können in den Rasterabbildungen dagegen nicht erfasst werden. Klarer festzustellen, sind die Amplituden der Simulation. Je mehr Stützstellen für die Integration verwendet werden, umso eindeutiger erscheinen die Modellparameterwerte in den Rastern. Dies zeigen die Rasterbeispiele der simulierten Daten der Lorentzfunktion mit drei Modellparametern, die mit und ohne angepasste Amplitudenpräzision ausgewertet wurden. In den Rastern der nicht angepassten, größeren Stützstellenanzahl (Abb. 4.11) konzentrieren sich die Rasterunterteilungen auf die Amplitudenlage eines Peaks. Die Raster der unteren beiden Graphen in Abbildung 4.14, die weniger Stützstellen zur Verfügung hatten, weisen auf zwei Amplitudenlagen hin. In den abgebildeten Rastern repräsentiert jede Dimensions-Achse einen Modellparameter eines Peaks. Somit können die Modellparameterwerte einzelner Peaks in den Rastern der nicht angepassten Auswertung eindeutiger beobachtet werden.
Für eine mögliche Abschätzung der Modellparameterwerte sollten nur Raster beachtet werden, die zur Integration für die Posterior-Berechnung der letztendlich wahrscheinlichsten Peakanzahl verwendet wurden.

In Tabelle 4.3 sind die Durchschnittswerte der Posterior-Wahrscheinlichkeiten für 3, 4 und 6 Peaks größer als der entsprechende Wert der wahren Peakanzahl. Dies liegt daran, dass die Auswertung der simulierten Daten viele Stützstellen und somit in den Evaluationsergebnissen ein paar Ausreißer hat. Die Werte dieser Ausreißer sind in dem dargestellten Beispiel so groß, dass sie den Mittelwert stark abändern. Ich habe die Durchschnittswerte zur Veranschaulichung der numerischen Werte der Ergebnisse berechnet. Möglicherweise ist es angebrachter, Medianwerte zu verwenden, da diese nicht zu sehr von Ausreißer-Werten beeinflusst werden.

Das Endergebnis beider Anwendungen mit simulierten Daten, die mit der Lorentzfunktion dreier Modellparameter erstellt wurden, ist das gleiche (Abb. 4.9 und Abb. 4.12). Die wahrscheinlichste Peakanzahl $N = 2$ wird in der Auswertung, die die angepasste Amplitudenpräzision anwendet, durch eine Evaluation mehr belegt als in der unangepassten Auswertung.
In Abbildung 4.12 wirkt das Ergebnis noch eindeutiger, da die übrigen Peakzahlen nur ein bis zwei Mal den maximalen Posterior-Wert erreichten. Abbildung 4.9 der Anwendung, die mehr Stützstellen verwendet, zeigt meines Erachtens einen nachvollziehbaren Verlauf der Ergebniswerte. Darin ist die zweit-wahrscheinlichste Peakanzahl, die auf die bestimmte Zahl folgende, $N = 3$. Den Ergebniswert für$N = 6$ betrachte ich als Ausreißer. Dies bestätigt sich durch den letzten Graphen der Abbildung 4.10, der ein exemplarisches Evaluationsergebnis zeigt, in dem die wahrscheinlichste Peakanzahl $N = 6$ ist.
Um den richtigen Schluss aus den durchgeführten Berechnungen ziehen zu können, ist es wichtig, das Endergebnis unter Berücksichtigung der Ergebnisse der einzelnen Evaluationen auszuwerten.

5.2.3. Auswertung der Messdaten

Meine Anwendung für gemessene Daten ermöglicht eine Unterteilung des Spektrums.
Auf diese Weise können enthaltene Multipletts einzeln betrachtet werden. Die Rechenzeit verkürzt sich dadurch gleichzeitig, da für den betreffenden Bereich die Wahrscheinlichkeiten weniger Peaks bestimmt werden. Die Dimension des multiplen Integrals in der Wahrscheinlichkeitsberechnung wird verringert und somit die Performanz gesteigert.
Dem Benutzer wird durch die Einteilungsmöglichkeit die Verantwortung übergeben, dies sachgerecht für die Multiplett-Analyse durchzuführen. Die Unterteilung des Spektrumsignals zur Auswertung eines Multipletts sollte an Stellen mit sehr geringer Auslenkung vorgenommen werden. Meine Anwendung überprüft die vorgenommene Einteilung nicht.
Ich habe die Einteilung dafür vorgesehen, einen Signalteil getrennt vom Restsignal auswerten zu können. Wenn ein Signalteil jedoch aus einem zusammenhängenden Multiplett genommen wird, kann es zu fehlerhaften Auswertungen kommen.

Bei der Eingabe von bereits geschätzten Werten in das Curve-Fitting-Tool habe ich nur den Startwert auf den Schätzwert gesetzt. Die Anpassung erschien mir etwas genauer, als ich den Minimum- und Maximumwert leicht abweichend zum Schätzwert angab. Dies hängt vermutlich damit zusammen, dass es sich um geschätzte Werte und nicht um wirklich feste oder gar die "wahren" Werte handelt. Ein kleiner Spielraum für die Anpassung des Tools ist daher vorteilhaft.

Wenn der Mischanteil (η) des Pseudo-Voigt-Profils nicht bereits bei der Messung durch Kalibrierung bestimmt wurde, kann er, wie in Abschnitt 3.3.3 erwähnt, geschätzt werden.
Es ist ebenso möglich η gemeinsam mit den Modellparametern zu marginalisieren, was ich später in einer Anwendung mit der VEGAS-Integration parallel umgesetzt habe. Ich habe mich anfangs für die Schätzung des η-Wertes mit dem Curve-Fitting-Tool entschieden, um das multiple Integral in P(N| D, I) nicht um eine Dimension zu vergrößern. Gleichzeitig gibt das Curve-Fitting-Tool die Möglichkeit, die Anpassung der Modellfunktion an die Messdaten und mögliche Schätzwerte der Modellparameter zu sehen.
Durch die Verwendung des Curve-Fitting-Tools sowie das Ablesen der Minima- und Maxima-Werte für die Modellparameter aus dem Graphen der Messdaten betrachtet der Benutzer die gemessenen Daten in einem Graphen. Wenn die reinen Zahlenwerte beobachtet werden, kann der Bezug zum Gegenstand der Auswertung zu sehr in den Hintergrund geraten.
Dies habe ich während der Erarbeitung der Diplomarbeit selbst erfahren. Evaluationswerte können durch die Anwendung beispielsweise genauer berechnet werden, als es grundsätzlich durch die Messung möglich ist. Die mit den Messdaten berechneten Werte dürfen und können sachgemäß nicht präziser sein als die Messwerte selbst. Der Benutzer beurteilt daher, ob die Präzision der Berechnungen angemessen ist. Das Betrachten der Messdaten in einem entsprechenden Graphen ist dafür teilweise geeigneter als sich die einzelnen Messwerte numerisch anzuschauen.
Auch das Ablesen einiger Werte aus den Graphen kann vorteilhafter sein, als sie von der Anwendung automatisch berechnen zu lassen. Der Benutzer weiß gewöhnlich besser, was er beachten muss, um den Wert im Graphen abzuschätzen. Der Anwendung müsste dieses Wissen in Form von Bedingungen eingegeben werden. Die Definition einer oder auch mehrerer Bedingungen kann jedoch nicht immer alle Eventualitäten, die auftreten können, abdecken.

Die erzielten Werte der χ^2-Minimierung in der Anwendung mit der numerischen Integration über die Likelihood-Matrix (*siehe Abb. 4.20*) waren nicht zufriedenstellend. Die Maximumwerte der Likelihood-Matrix waren in den Graphen der einzelnen Modellparameter in Abbildung 4.19 auch nach der Anpassung der Amplitudenpräzision für die Stützwerte der Matrix nicht eindeutig genug. Die beiden Graphen des dritten Peaks zeigen im Gegensatz zu denen der ersten beiden Peaks keine Veränderung durch die Präzisionsanpassung.
Dennoch half mir die Anwendung weiter, eine Vermutung über das Fehlschlagen der Anwendung mit

dem VEGAS-Algorithmus aufzustellen.
Bei den Messdaten sind die Abstände der Messpunkte entlang der Abszisse bei gleichmäßiger Abtastung einheitlich. In Richtung der Signalintensität variieren die Abstände natürlicherweise. Die Abstände der Messpunkte in y-Richtung sind an den Stellen groß, an denen das Spektrum steil verläuft. In diesen Bereichen befinden sich hohe Intensitätswerte. Die Amplituden der Messdaten stellen meist eine Spitze dar und sind somit je nach Höhe und Abtastrate durch relativ wenige Messpunkte gegeben.
Der VEGAS-Algorithmus setzt mittels des Importance Samplings mehr Stützstellen in Bereichen hoher Funktionswerte. Da die Messdaten dort aber nicht viele Werte aufweisen, kommt es vermutlich dazu, dass die Abstände zwischen den VEGAS-Stützstellen kleiner sind als die zwischen den Messpunkten in diesem Bereich.
Bei der Auswertung mit dem VEGAS-Algorithmus wird das Integrationsergebnis mit wachsender Dimension bzw. Peakanzahl kleiner, die Posterior- Wahrscheinlichkeit größer.
Ich habe versucht das Problem über eine angepasste Vorgabe der Stützstellenanzahl abzuschwächen. Durch die Anpassung der Stützstellenpräzision, beschrieben am Ende des Abschnittes 3.3.3, habe ich die Anzahl der Stützstellen verkleinert. Dies brachte jedoch kaum Erfolg. Statt nur eine Präzision für die Amplitudenstützwerte zu verwenden, habe ich die Anwendung auch mit zwei unterschiedlichen Stützwertabständen ausprobiert. Leider änderten sich die Ergebnisse dadurch nicht.

Bei der umgesetzten Anwendung mit der VEGAS-Integration ist in den Ergebnisgraphen der einzelnen Evaluationen zunächst eine Steigungsänderung der Verlaufskurve der logarithmischen Posterior-Werte zu erkennen (*siehe Abbildung 4.22*).
Dieser Knick in den Kurven zeigt an, bis zu welcher Peakanzahl das zugehörige Integral noch einen größeren Wert hatte als das definierte, kleinst-mögliche Ergebnis der VEGAS-Integration von $1 \cdot 10^{-16}$. Bei der Auswertung mit drei angegebenen Amplitudenlagen befindet sich die Änderung der Steigung im Ergebnisgraphen erst bei dem vierten Peak. Dies ist der Fall, da die Integrationen über weniger Dimensionen und dadurch mit kleinerer Stützstellenanzahl durchgeführt wurde als die beiden anderen Auswertungsvarianten. Aufgrund der Festlegung des kleinst-möglichen Integrations-Ergebnisses waren alle Integrale ab dem vierten bzw. fünften Peak gleich groß(*siehe Tabellen 4.5, 4.6 und 4.7*). Dadurch war die Prior-Wahrscheinlichkeit der ausschlaggebende Faktor für die maximale Posterior-Wahrscheinlichkeit.
Die Posterior-Wahrscheinlichkeiten für die Evaluationen ohne vorgegebene Amplitudenlagen waren übereinstimmend, da die berechneten Integrale der Peakzahlen beider Anwendungen annähernd gleich waren. Dies ist in den Tabellen 4.6 und 4.7 zu sehen. Der Prior der jeweiligen Peakanzahl war für die beiden Evaluationsvarianten gleich, da die zusätzliche Integration über η ihn nicht verändert, solange die Grenzen für den η-Schätzwert auf 0 und 1 gesetzt sind. Bei der Evaluation, die drei Schätzwerte für die Amplitudenlage verwendete, ist der Prior kleiner, da über drei Modellparameter weniger integriert wird. Aus diesem Grund sind die Werte ihrer Posterior-Wahrscheinlichkeiten nicht so groß wie die der zwei anderen Evaluationsarten.

Fast alle Posterior-Wahrscheinlichkeiten sind größer als eins und liegen somit auf der logarithmischen Skala im positiven Wertebereich. Die maximale Wahrscheinlichkeit bzw. die Summe aller möglichen Wahrscheinlichkeiten einer Beobachtung ist definitionsgemäß eins.
Da ich bei der Wahrscheinlichkeitsberechnung einige Skalierungskonstanten ausgelassen habe (*siehe Abschnitt 3.1*), können die errechneten Posterior-Wahrscheinlichkeiten höhere Werte aufweisen.
Je mehr Peaks verwendet werden, umso wahrscheinlicher scheint ihre Anzahl für das gemessene Spektrum. Dies sollte bei dieser Arbeit nicht der Fall sein. Das Ziel war eine endliche Peakanzahl zu finden, die für die gemessenen Daten am wahrscheinlichsten ist.
Dass es nicht zu dem angestrebten und aussagekräftigen Ergebnis kommt, könnte u. a. an der Abhängigkeit der Stützstellenanzahl von den Integrationsdimensionen liegen. Ich habe die Vorgabe für die Stützstellenzahl *ncall* über die Messwert-Präzision der einzelnen Dimensionen berechnet. Die geeignete *ncall*-Angabe ist experimentell für die Integration zu bestimmen, wie ich in Grundlagen-

abschnitt 2.5 beschrieben habe. Anfangs hatte ich die Angabe für die Anzahl der Stützstellen frei gewählt und sie für alle Posterior-Berechnungen gleich gesetzt. Die Posterior-Werte stiegen dadurch mit der Peakanzahl exponentiell. Über die Anpassung des Stützstellenabstandes habe ich versucht, *ncall* einzugrenzen. Dadurch wurde es jedoch proportional zur Dimension, da ich die Vorgabe der Stützstellenanzahl für jede Dimensions-Achse berechne und summiere. Die Ergebnisse zeigen, dass *ncall* für die höheren Dimensionen zu groß gewählt ist. Für die Integration über wenige Dimensionen ist es dagegen nicht ausreichend, wie die mehrfachen Berechnungen für das optimale Raster in Ergebnisabschnitt 4.2.2 zeigen. Ich habe die geeignete Stützstellenvorgabe für meine Anwendung nicht gefunden.
Eine mögliche Änderung ist, die *ncall* für die genauere Integration nicht zehnmal so groß anzugeben wie die Stützstellenanzahl für die Rasterbestimmung. Der Faktor könnte beispielsweise umgekehrt proportional zur Integrationsdimension gewählt werden.
Dabei wirkt sich auf diese Art die Dimensionsabhängigkeit voraussichtlich nicht nachteilig aus.

Die unzureichenden Ergebnisse könnten auch mit der Schätzung des Mischanteils η zusammenhängen. Im Curve-Fitting-Tool habe ich η für die möglichst genaue Anpassung an die Messdaten bestimmt. Diese verbesserte sich mit steigender Peakanzahl.
Der angenommene Wert von η sollte für die unterschiedlichen Peakzahlen nicht zu sehr variieren, da der wahre Wert für alle Peaks einer Messung gleich ist. Die drei geschätzten Amplitudenlagen
wurden durch ein anderes Verfahren der Bayes'schen Datenanalyse mit einem η von etwa 0,3 ermittelt. Dagegen liegen meine geschätzten η-Werte zwischen 0,5 und 0,6.
Es ist einfacher, den kalibrierten Wert für η zusammen mit den Messdaten festzustellen, anstatt ihn über die Messdaten zu schätzen bzw. ihn zu marginalisieren. Auf diese Weise ist das Mischverhältnis für die Auswertung vorgegeben und einheitlich für alle möglichen Spektren unterschiedlicher Peakzahl.

6. Schlussfolgerung und Ausblick

Im Rahmen meiner Diplomarbeit gelang es nicht, die Bestimmung der Peakanzahl in einem gemessenen Spektrum erfolgreich zu klären.
Ich habe meine Auswertung auf unterschiedliche Arten und Weisen modifiziert und dadurch teilweise verbessern können. Meine Vermutung ist, dass die Schwierigkeit, plausible Ergebnisse zu erzielen an der numerischen Integration in Verbindung mit Messdaten und der Wahl des Stützstellenabstandes in der Dimension liegt, die nicht der Kontrollvariablen in x-Richtung entspricht.
Bei der Integrationsmethode mit der Likelihood-Matrix ist der Abstand der Stützstellen über die Schrittgröße der Modellparameterwerte, mit der die Stützstellen und die Matrix erstellt werden, festlegbar. Für jede Dimension bzw. jeden einzelnen Modellparameter ist dies individuell möglich. Die Integration über die Likelihood-Matrix dauerte im Vergleich zur VEGAS-Integration äußerst lange (*siehe Ergebnisabschnitt 4.2.2*) und führte im Rahmen meiner Diplomarbeit nicht zum Ziel.
Unter Verwendung des VEGAS-Algorithmus ergibt die multidimensionale Integration, deren Integrand einen gemessenen Datensatz enthält, keine plausiblen Werte.
Das berechnete Integral wächst mit den Integrationsdimensionen bzw. der Peakanzahl.
Das Importance Sampling des Algorithmus wirkt sich durch die Konzentration der Stützstellen auf Bereiche hoher Funktionswerte anscheinend nachteilig auf diese Anwendung aus.
Im Diskussionskapitel habe ich ein paar alternative Umsetzungselemente beschrieben, die das Resultat möglicherweise verbessern könnten.

Für den Prior $P(\{A_j, \epsilon_j, w_j\}|N, I)$ habe ich eine gleichmäßige Verteilung angenommen. Die Werte der verschiedenen Modellparameter können unterschiedlich verteilt sein.
Mit weiterführenden, analytischen Überlegungen könnten realistischere, nicht unbedingt gleichmäßige Prior-Wahrscheinlichkeitsverteilungen festgelegt werden.

Die Bestimmung der Peakanzahl eines Spektrums kann auch für Analysen anderer Anwendungsgebiete der Spektroskopie nützlich sein. Messdaten der Elektrophorese werden zum Beispiel in Form eines zweidimensionalen Spektrums ausgewertet. Die Abszisse bildet dabei die Distanz ab, die die Probe in dem verwendeten Messmedium zurückgelegt hat. Die Intensität der Moleküle an einer Stelle wird durch die Auslenkung der Kurve angezeigt.

Um einem Benutzer meine erstellte Anwendung zugänglicher zu machen, könnte eine benutzerfreundliche Oberfläche gestaltet werden. Auf diese Weise müsste der Benutzer keine Werte in die Matlab-Konsole eingeben und die Ergebnisse würden in einer einheitlichen Umgebung angezeigt.

Mit einer Anwendung, die zuverlässige Ergebnisse über die Peakanzahl liefert, wäre es möglich das gemessene Spektrum anzunähern. Über die Verknüpfung mit einer Datenbank, die zuvor identifizierte Spektren beinhaltet, könnte das bestimmte Spektrum mit diesen abgeglichen und ausgewertet werden.

Meine erstellte Arbeit führt leider nicht zu dem angestrebten Ziel. Ich hoffe dennoch, dass sie für weitere Forschungsarbeiten zur quantitativen Analyse von Spektren hilfreich ist.
Zum Beispiel könnte die Anwendung für real gemessene Daten so abgeändert werden, dass man mit ihr die Wahrscheinlichkeiten unterschiedlicher Peakanzahlen für ein Spektrum vergleichen kann. Dies wird möglich, wenn die Modellparameterwerte der einzelnen Peaks bereits geschätzt sind. Das Mischverhältnis η des Pseudo-Voigt-Profils sollte dafür während der Messung kalibriert worden sein

und somit für alle Schätzungen einheitlich gelten. Die Posterior-Wahrscheinlichkeiten können mit Hilfe der geschätzten Werte aller Modellparameter ohne Marginalisierung berechnet werden.
Auf diese Weise könnten Ergebnisse aus anderen Bayes' basierten Verfahren zur Bestimmung der Peakanzahl bewertet und ihre Plausibilität quantitativ überprüft werden. Zudem ist mit dem Curve-Fitting-Tool und den Funktionen, die Spektren verschiedener Peakanzahlen darstellen, die Übereinstimmung der Messdaten gegenüber einem angenäherten Spektrum über deren gemeinsame Anzeige in einem Graphen visuell beurteilbar.

Literaturverzeichnis

[1] Peter M. Skrabal. *Spectroscopy: An interdisciplinary integral description of spectroscopy from UV to NMR*. vdf Hochschulverlag AG, 2012.

[2] Christian Gerthsen. *Physik: Ein Lehrbuch zu Gebrauch neben Vorlesungen*. Springer-Verlag, 9 edition, 2013.

[3] Mike Johns, Einar O. Fridjonsson, Sarah Vogt, and Agnes Haber. *Mobile NMR and MRI: Developments and Applications*. The Royal Society of Chemistry, 2016.

[4] D. S. Sivia and C. J. Carlile. Molecular spectroscopy and bayesian spectral analysis - how many lines are there? *The Journal of Chemical Physics*, 96, 1992.

[5] Kernspinresonanzspektroskopie. Website: `https://de.wikipedia.org/wiki/Kernspinresonanzspektroskopie`, 2016.

[6] M. Hesse, H. Meier, and B. Zeeh. *Spektroskopische Methoden in der organischen Chemie*. Thieme Verlag, 7 edition, 2005.

[7] H. Lohninger, J. Fröhlich, B. Mizaikoff, and E. Rosenberg. *Teach/Me Instrumentelle Analytik*, chapter Elektromagnetsysteme. Springer-Verlag, 2010. Online unter: `http://www.vias.org/tmanalytik_germ/hf_nmr_technik_elektromagnetsysteme.html`.

[8] J. P. Hornak. Cw nmr experiment. Website: `http://www.cis.rit.edu/htbooks/nmr/inside.htm`, 2011.

[9] Science - introduction to nuclear magnetic resonance. Website: `http://www.magritek.com/support/science/`, 2015.

[10] Xiaobo Qu, Di Guo, Xue Cao, Shuhui Cai, and Zhong Chen. Reconstruction of self-sparse 2d nmr spectra from undersampled data in the indirect dimension. *Sensors*, 11, 2011. Online unter: `http://www.quxiaobo.org/project/self-sparseNMR/`.

[11] H. Lohninger, J. Fröhlich, B. Mizaikoff, and E. Rosenberg. *Teach/Me Instrumentelle Analytik*, chapter Die Fouriertransformation. Springer-Verlag, 2010. Online unter: `http://www.vias.org/tmanalytik_germ/hf_nmr_spektren_fouriertransformation.html`.

[12] L. Ernst. *13C-NMR-Spektroskopie: Eine Einführung*. Dr. Dietrich Steinkopff Verlag GmbH, 1980.

[13] D. S. Sivia and J. Skilling. *Data Analysis - A Bayesian Tutorial*, chapter The basics, Probability: Cox and the rules for consistent reasoning, pages 4 – 5. Oxford University Press Inc., New York, 2 edition, 2006.

[14] D. S. Sivia and J. Skilling. *Data Analysis - A Bayesian Tutorial*, chapter The basics, Corollaries: Bayes' theorem and marginalization, pages 5 – 8. Oxford University Press Inc., New York, 2 edition, 2006.

[15] D. S. Sivia and J. Skilling. *Data Analysis - A Bayesian Tutorial*, chapter Model selection, Example 6: how many lines are there?, pages 85 – 93. Oxford University Press Inc., New York, 2 edition, 2006.

[16] E. T. Jaynes. *Papers on Probability, Statistics, and Statistical Physics*. Reidel, 1983.

[17] E. T. Jaynes. *Maximum Entropy and Bayesian Methods in Applied Statistics*. Cambridge University, 1986.

[18] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27, 1948.

[19] J. E. Shore and R. W. Johnson. Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy. *IEEE Transactions on Information Theory*, 26, 1980.

[20] Y. Tikochinsky, N. Z. Tishby, and R. D. Levine. Consistent inference of probabilities for reproducible experiments. *Physical Review Letters*, 52, 1984.

[21] J. Skilling and S. F. Gull. *Maximum Entropy and Bayesian Methods in Science and Engineering*. Kluwer Academic Publishers, 1988.

[22] J. Skilling and S. F. Gull. *Maximum Entropy and Bayesian Methods*. Kluwer Academic Publishers, 1989.

[23] S. F. Gull and J. Skilling. Maximum entropy method in image processing. *Communications, Radar and Signal Processing, IEE Proceedings F*, 131, 1984. and references therein.

[24] H. Jeffreys. *Theory of Probability*. Kluwer Academic Publishers, 1939.

[25] G. L. Bretthorst. *Bayesian Spectrum Analysis and Parameter Estimation*. Springer-Verlag GmbH, 1988.

[26] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7, 1965.

[27] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes. The Art of Scientific Computing*. Cambridge University Press, 3 edition, 2007.

[28] R. Mukhopadhyay, C.J. Carlile, and R. N. Silver. Application of maximum entropy to neutron tunneling spectroscopy. *Physica B*, 174, 1991.

[29] Ludwig Fahrmeir, Christian Heumann, Rita Künstler, Iris Pigeot, and Gerhard Tutz. *Statistik: Der Weg zur Datenanalyse*. Springer-Verlag GmbH, Berlin - Heidelberg, 8 edition, 2016.

[30] G. Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27, 1978.

[31] G. Peter Lepage. Vegas: An adaptive multidimensional integration program. 1980. CLNS-80/447.

[32] Voigt-profil. Website: `https://de.wikipedia.org/wiki/Voigt-Profil`, 2016.

[33] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes. The Art of Scientific Computing*, chapter Adaptive Monte Carlo: VEGAS, pages 414 – 416. Cambridge University Press, New York, 3 edition, 2007.

[34] Sonja Surjanovic and Derek Bingham. Virtual library of simulation experiments: Test functions and datasets, optimization test problems - rosenbrock function. Website: `https://www.sfu.ca/~ssurjano/rosen.html`, 2015.

[35] W. H. Heini Gränicher. *Messung beendet - was nun?: Einführung und Nachschlagewerk für die Planung und Auswertung von Messungen*. vdf Hochschulverlag AG, 2 edition, 1996.

[36] Faddeeva function. Website: https://en.wikipedia.org/wiki/Faddeeva_function, 2016.

[37] Curve fitting toolbox. Website: https://de.mathworks.com/products/curvefitting/?requestedDomain=de.mathworks.com, 2016.

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

FID Free Induction Decay

FWHM Full Width Half Maximum, Halbwertsbreite

NMR Nuclear Magnetic Resonance

NR Numerical Recipes

Anhang

A. VEGAS-Variablen

mds int	gibt den Sampling-Modus an: • mds = -1 : Anwendung von Importance und Stratified Sampling. Rastereinteilungen konzentriert an der Stelle, an der der Fehler am größten ist. • mds = 0 : nur Anwendung des Importance Samplings • mds = 1 : Anwendung von Importance und Stratified Sampling. Rastereinteilungen konzentriert an der Stelle, an der der Integrand am größten ist.
ndo int	bisherige Anzahl der Rastereinteilungen entlang einer Dimensions-Achse (nd 'old' -> Vergleich mit *nd*, um zu sehen, ob Raster für Integration angepasst werden muss)
xi mat(j, i) double	Lage der i-ten Rastereinteilung auf der j-ten Dimensions-Achse normalisiert auf Werte [0..1]
si double	$\sum I_i / \sigma_i^2$
swgt double	$\sum 1 / \sigma_i^2$
schi double	$\sum I_i^2 / \sigma_i^2$
tgral double	Beste Schätzung für Integral (I_{best})
chi2a double	Chi^2 pro Freiheitsgrad $(\sum I_i^2 / \sigma_i^2 - \sum I_i / \sigma_i^2 * I_{best}) / (it - 0.9999)$
sd double	Standardabweichung des besten Integralschätzers (σ_{best})
nd int = *xnd* double	Anzahl der Rastereinteilungen entlang einer Dimensions-Achse
ng int	Anzahl der Rastereinteilungen auf einer Dimensionsachse
npg int	Anzahl der Stützstellen innerhalb eines Hypercubes (>=2)
k int	Vor Hauptiterationsschleife: Anzahl der Hypercubes im Volumen danach: Laufvariable für Stützstellen eines Hypercubes
calls double	Anzahl der Stützstellen, die in den Integrationsiterationen verwendet werden
dxg double	=1/ng → Berechnung von dv2g; danach → Berechnung von xn für Zufallspunktberechnung: = nd/ng
dv2g double	Varianz generierter Stützstellen eines Cubes → Berechnung der Varianz einer Integration
dx vec(dim) double	Maße des Volumens (regn), über das integriert wird, für jeweilige Dimension
xjac double	Verhältnis der Volumengröße, über das integriert wird, zur Anzahl der Integrationspunkte (→ wgt)
r vec (Raster-unter-teilungen) double	unterteilt Rasterabschnitte, verwendet für rebin-Funktion; kontrolliert die Rasterverfeinerung, u.a. auch den Umfang/ Geschwindigkeit, in dem Raster verändert wird (→ ALPHA)

xin vec (Raster- unter- teilungen) double	Hilfsvektor für Rasterveränderungen in rebin-Funktion
ti double	geschätztes Integral einer Iteration (I_i)
tsi double	Varianz (σ_i^2) bzw. Standardabweichung (σ_i) von ti (I_i) (=∑f2b aller Rastercubes * dv2g)
kg vec(dim) int	→ Generierung der Stützstellen
d mat(Raster- unter- teilungen, dim) double	mds >= 0 : =$\sum f^2$; mds = -1 : =∑f2b $f_i \Delta x_i$
di mat(Raster- unter- teilungen, dim) double	∑ΔI = I; Anteile der Rastereinteilungen an gesamtem Integral di(i, j) := Summe aller *f*-Werte in der i-ten Rasterunterteilung auf der j-ten Dimensionsachse
fb double	∑f der Zufallspunkte innerhalb eines Rastercubes; f_i
f2b double	$\sum f^2$ der Zufallspunkte innerhalb eines Rastercubes; danach weiter verändert, zur Berechnung der Varianz eines Integrals: $(\sum f^2(x_i)/p(x_i)) - I_i^2$
wgt double	Gewichtung für Integrationspunkt $f(x_i)$; danach zur Berechnung der Endergebnisse: $=1/\sigma_i^2$
xn double	Rechenvariable: → Berechnung der Zufallspunkte; → Rasterverfeinerung nach Integrationsberechnung: auf xo folgender Wert in *d*-Matrix
ia vec(dim) int	Indices der Rasterunterteilungen 1<= ia(dim) <= nd <= NDMX
xo double	Rechenvariable:→ Berechnung der Zufallspunkte: Länge der Rastereinteilung in einer Dimension; → Rasterverfeinerung nach Integrationsberechnung: Wert in *d*-Matrix
rc double	→ rebin: in Vorbereitung auf Integration: Verhältnis der Anzahl der alten Rastereinteilungen zu der Anzahl neuer Rastereinteilungen nach jeder Integration: Verhältnis der Summe aller Werte in *r* zur Anzahl der Rastereinteilungen; (sollte Werte zwischen 0 und 1 annehmen, rc > 1 → Stützstellen außerhalb von regn) → Berechnung der Zufallspunkte
f double	gewichteter Funktionswert für Zufallspunkt
x vec(dim) double	Zufallspunkt innerhalb des Volumens, über das integriert wird
f2 double	f^2
dt vec(dim) double	Summe der Werte in *d* einer Dimension $\sum f_i \Delta x_i$

B. Implementierung in Matlab

B.1. VEGAS-Code

```
function [tgral, sd, chi2a] = vegas(regn, fxn, init, ncall, itmx,...
    nprn, prfp, nmp, rosenb)
% vegas: Performs Monte Carlo integration.
%
%   tgral  - best estimate of integral
%   sd     - standard deviation of best integral estimate
%   chi2a  - chi^2 per degree of freedom,
%            indicator of whether consistent results are being obtained
%
%   regn   - array defining volume over that fxn is integrated, contains
%            2*dim coordinates, "downer left" followed by "upper right"
%   fxn    - integrand function
%   init   - flag to signal whether this call is a new start,
%            or a subsequent call for additional iterations
%   ncall  - number of random points for estimation of integral /
%                number of calls to function fxn
%   itmx   - number of iterations
%   nprn   - controls amount of diagnostic output (normally 0)
%   prfp   - flag to signal whether to print result integral in fixed-point
%            or more compact notation
%   nmp    - number of model parameters of integrand
%   rosenb - flag to signal whether integration is for rosenbrock function
%            or to estimate number of peaks; used for grid plot
%-------------------------------------------------------------------------

    % controlls rate at that grid is modified from iteration to iteration
    ALPHA = 1.5;

    % maximum number of increments along each axis
    NDMX = 50;

    % smallest value for results
    TINY = 1.0e-16;

    %integer values
    persistent i it j k mds nd ndo ng npg

    %integer arrays
    persistent ia kg

    %double values
    persistent calls dv2g dxg f f2 f2b fb rc ti tsi wgt xjac xn xnd xo
    persistent schi si swgt
```

```
%double arrays
persistent dt dx r x xin

%double matrices
persistent xi d di

% dimension of integral
ndim = int32(numel(regn)/2 -0.5);

% Allocation of persistent arrays and matrices
if(init == 0 || isempty(ia))
    ia = zeros(1, ndim, 'int32');
end
if(init == 0 || isempty(kg))
    kg = zeros(1, ndim, 'int32');
end
if(init == 0 || isempty(d))
    d = zeros(NDMX, ndim, 'double');
end
if nprn && (init == 0 || isempty(di))
    di = zeros(NDMX, ndim, 'double');
end
if(init == 0 || isempty(dt))
    dt = zeros(1, ndim, 'double');
end
if(init == 0 || isempty(dx))
    dx = zeros(1, ndim, 'double');
end
if(init == 0 || isempty(r))
    r = zeros(1, NDMX, 'double');
end
if(init == 0 || isempty(x))
    x = zeros(1, ndim, 'double');
end
if(init == 0 || isempty(xi))
    xi = zeros(ndim, NDMX, 'double');
end
if(init == 0 || isempty(xin))
    xin = zeros(1, NDMX, 'double');
end

% Normal entry. Enter here on a cold start.
if (init <= 0)
    mds = int32(1);  % Change to mds=0 to disable stratified sampling
                     % i.e., use importance sampling only.
    ndo = int32(1);

    % one big hypercube, no grid yet
    xi(1:ndim, 1) = 1.0;
end
```

```
% Enter here to inherit the grid from a previous call,
% but not its answers
if (init <= 1)
   si   = 0.0;
   swgt = 0.0;
   schi = 0.0;
end

% Enter here to inherit the previous grid and its answers.
if (init <= 2)
    nd = NDMX;
    ng = int32(1);

    % Set up for stratification
    if (mds)
        ng = int32((double(ncall)/2.0 + 0.25)^(1.0/double(ndim))-0.5);
        mds = int32(1);
        if ((2*ng - NDMX) >= 0) % 2*ng >= nd
            mds = int32(-1);
            npg = int32(double(ng)/double(NDMX) + 1 -0.5);
            nd = int32(double(ng)/double(npg) -0.5);
            ng = npg*nd;
        end
    end
    %number of hypercubes in volume
    k = ng^ndim;
    npg = max(int32(double(ncall)/double(k) -0.5), 2);
    calls = double(npg)*double(k);
    dxg = 1.0/double(ng);
    dv2g = dxg^double(ndim);
    dv2g = (calls * dv2g)^double(2)/ double(npg)/...
        double(npg)/(double(npg)-1.0);

    xnd = double(nd);
    dxg = dxg * xnd;
    xjac = 1.0/calls;
    for j=1:ndim
        dx(j) = regn(j + ndim) - regn(j);
        xjac = xjac * dx(j);
    end

    % Do binning if necessary, create first grid
    if (nd ~= ndo)
        r(1:max(nd,ndo))=1.0;
        for j=1:ndim
            xi(j,:) = rebin(double(ndo)/xnd, nd, r, xin, xi(j,:));
        end
        ndo = nd;
    end
    % print integration parameters
    if (nprn >= 0)
        fprintf('%s: ndim= %3d ncall= %8.0f\n',...
```

```
            'Input␣parameters␣for␣vegas', ndim, calls);
        fprintf('%28s␣it=%5d␣itmx=%5d\n','␣', it, itmx);
        fprintf('%28s␣nprn=%3d␣ALPHA=%5.2f\n','␣', nprn, ALPHA);
        fprintf('%28s␣mds=%3d␣nd=%4d\n','␣', mds, nd);
        for j=1:ndim
            fprintf('%30s␣xl[%2d]=␣%11.4g␣xu[%2d]=␣%11.4g\n',...
                '␣', j, regn(j), j, regn(j+ndim));
        end
    end
end

% Main iteration loop. Can enter here (init >= 3)to do additional itmx
% iterations with all other parameters unchanged.
for it=1:itmx

    % print optimal grid (init = 1)
    % that is used to calculate integral of optional dimension (ndim)
    if(init == 1 && (ndim == 9))
        printGrid(nmp, rosenb, ndim, nd, xi, regn);
    end

    ti = 0.0;
    tsi = 0.0;
    kg(1:ndim) = int32(1);
    d(1:nd, 1:ndim) = 0.0;
    if(nprn)
        di(1:nd, 1:ndim)= 0.0;
    end

    while 1
        fb = 0.0;
        f2b = 0.0;
        for k=1:npg
            wgt = xjac;
            for j=1:ndim
                random = rand;
                xn = (double(kg(j)) - random)*dxg + 1.0;
                ia(j) = max( min(int32(xn -0.5), NDMX), 1);
                if (ia(j) > 1)
                    xo = xi(j, ia(j)) - xi(j, ia(j)-1);
                    rc = xi(j, ia(j)-1)+(xn - double(ia(j)))*xo;
                else %ia(j)=1
                    xo = xi(j, ia(j));
                    rc = (xn - double(ia(j)))*xo;
                end
                x(j) = regn(j) + rc*dx(j);
                wgt = wgt * xo * xnd;
            end %for(dimensions)

            f = wgt * double(fxn(x));

            f2 = f*f;
```

```
            fb = fb + f;
            f2b = f2b + f2;
            for j=1:ndim
                if(nprn)
                    di(ia(j), j) = di(ia(j), j) + f;
                end
                if (mds >= 0)
                    d(ia(j), j) = d(ia(j), j) + f2;
                end
            end
        end %for(npg/ number of sample points in one hypercube)
        f2b = sqrt(f2b*double(npg));
        f2b = (f2b - fb)*(f2b + fb);
        if (f2b <= 0.0)
            f2b = TINY;
        end
        ti = ti + fb;
        tsi = tsi + f2b;

        %Use stratified sampling
        if (mds < 0)
            for j=1:ndim
                d(ia(j), j) = d(ia(j), j) + f2b;
            end
        end
        for k = ndim : -1 : 0
            if(k > 0)
                kg(k) = mod(kg(k), ng);
                kg(k) = kg(k) + 1;
                if (kg(k) ~= 1)
                    break;
                end
            end
        end
        if (k < 1)
            break;
        end
    end %while(1)

    % Compute final results for this iteration.
    tsi = tsi * dv2g;
    if(tsi ~= 0)
        wgt = 1.0 /tsi;
    else
        wgt = 0;
    end
    si = si + wgt*ti;
    schi = schi + wgt*(ti^2);
    swgt = swgt + wgt;
    if(swgt ~= 0)
        tgral = si/swgt;
    else
```

```
        tgral = 0;
    end
    if(rosenb && tgral > 0 && tgral < TINY)
        tgral = TINY;
    end
    chi2a = (schi - si*tgral)/(double(it)-0.9999);
    if (chi2a < 0.0)
        chi2a = 0.0;
    elseif(chi2a > 0 && chi2a < TINY)
        chi2a = TINY;
    end
    sd = sqrt(1.0/swgt);
    tsi = sqrt(tsi);
    if (nprn >= 0)
        if(prfp)
            fprintf('%s␣%3d␣:␣integral␣=␣%14.2f␣+/-␣%9.2f\n',...
                '␣iteration␣no.␣', it, ti, tsi);
            fprintf('%s␣%14.2f␣+/-␣%9.2f␣chi^2/IT␣n␣=␣%9.2g\n',...
                '␣all␣iterations:␣integral␣=␣', tgral, sd, chi2a);
        else
            fprintf('%s␣%3d␣:␣integral␣=␣%14.7g␣+/-␣%9.2g\n',...
                '␣iteration␣no.␣', it, ti, tsi);
            fprintf('%s␣%14.7g␣+/-␣%9.2g␣chi^2/IT␣n␣=␣%9.2g\n',...
                '␣all␣iterations:␣integral␣=␣', tgral, sd, chi2a);
        end
        if (nprn)
            for j=1:ndim
                fprintf('␣DATA␣FOR␣axis␣%2d\n',j);
                fprintf('%6s%13s%11s%13s%11s%13s\n',...
                'X','delta␣i','X','delta␣i','X','delta␣i');
                for i=int32(1+nprn/2 -0.5) : nprn+2 : nd-2
                    fprintf('%8.5f%12.4g%12.5f%12.4g%12.5f%12.4g\n',...
                    xi(j, i), di(i, j), xi(j, i+1),...
                    di(i+1, j), xi(j, i+2), di(i+2, j));
                end
            end
        end %if(print grid data)
    end %if(print results)

    % Refine the grid. Consult references to understand
    % the subtlety of this procedure. The refinement is damped,
    % to avoid rapid, destabilizing changes,
    % and also compressed in range by the exponent ALPHA.
    for j=1:ndim
        xo = d(1, j);
        xn = d(2, j);
        d(1, j)=(xo + xn) /2.0;
        dt(j) = d(1, j);
        for i=2:nd-1
            rc = xo + xn;
            xo = xn;
            xn = d(i+1, j);
```

```
                d(i, j) = (rc+xn)/3.0;
                dt(j) = dt(j) + d(i, j);
            end
            d(nd, j)=(xo + xn) /2.0;
            dt(j) = dt(j) + d(nd, j);
        end %for(dimensions)
        for j=1:ndim
            rc = 0.0;
            for i=1:nd
                if (d(i, j) < TINY)
                    d(i, j) = TINY;
                end
                r(i) = ((1.0 - d(i, j)/dt(j))/ ...
                        (log(dt(j))-log(d(i, j))))^ALPHA;
                rc = rc + r(i);
            end
            rcp = rc/xnd;
            xi(j, :) = rebin(rcp, nd, r, xin, xi(j, :));
        end %for(dimensions)
    end %for(iterations itmx)
end

% Utility routine used by vegas, to rebin a vector of densities xivec into
% new bins defined by a vector rvec.
function xij = rebin(rc, nd, rvec, xinvec, xivec)
k = 0; dr = 0.0; xo = 0.0;
for i=1:nd-1
    while (rc > dr)
        k = k + 1;
        dr = dr + rvec(k);
    end
    if (k > 1)
        xo = xivec(k-1);
    end
    if(k > 0)
        xn = xivec(k);
        dr = dr - rc;
        xinvec(i) = xn - (xn - xo)*dr / rvec(k);
    end
end
for i=1:nd-1
    xivec(i) = xinvec(i);
end
xivec(nd) = 1.0;
xij = xivec;
end

function printGrid(nmp, rosenb, ndim, nd, xi, regn)
%
% nmp - number of model parameters
% rosenb - 1: rosenberg or 0: number pf peaks
% ndim - number of dimensions
```

```
% nd - number of increments along one axis
% xi - matrix of grid data per dimension axis
% regn - integration volume
%
if(rosenb)
    step = 1;
else
    step = nmp;
end

for dim = 1:step:ndim-1
    for gr = 1:nmp-1
        if(rosenb)
            dim1 = (['x' num2str(dim)]);
            dim2 = (['x' num2str(dim+gr)]);
        else
            ni = (dim-1)/nmp + 1;
            dim1 = (['x0_' num2str(ni)]);
            if(gr == 1)
                dim2 = (['w' num2str(ni)]);
            elseif(gr == 2)
                dim2 = (['A' num2str(ni)]);
            end
        end
        title = ([dim1 '-' dim2 '-Optimal grid for VEGAS algorithm']);
        figure('Name', title, 'NumberTitle', 'off');
        for g = 1:nd
            % x / first dimension axis grid lines
            x_val = xi(dim,g)*(regn(dim+ndim)-regn(dim))+regn(dim);
            line([x_val x_val], [regn(dim+gr) regn(dim+gr+ndim)],...
                'Color', 'k');

            % y / second dimension axis grid lines
            y_val = xi(dim+gr,g)*(regn(dim+gr+ndim)-regn(dim+gr))+...
                                                    regn(dim+gr);
            line([regn(dim) regn(dim+ndim)], [y_val y_val], 'Color', 'k');
        end
        xlabel([dim1 ' dimension']);
        ylabel([dim2 ' dimension']);
    end
end %for(dimensions)

end
```

B.2. Rosenbrock-Anwendung

```
function rosenbrock(d)
%
% d - dimension of rosenbrock function
%

nSamplingPts1 = 100000;
%input('Number of sampling points to find VEGAS grid: ');

nSamplingPts2 = 1000000;
%input('Number of sampling points to calculate integral: ');

if(d == 2)
    %integration limits
    min = -5; max = 10;
    regn = [min min max max];

    % plot integrand function
    x1 = linspace(-5,10);
    x2 = linspace(-5,10);
    [x1m,x2m] = meshgrid(x1,x2);
    z = (1-x1m).^2 +100*(x2m-x1m.^2).^2;
    figure('Name', '2D-Rosenbrock', 'NumberTitle', 'off');
    surf(x1, x2, z);caxis([-5,10])
    xlabel('x1 dimension');
    ylabel('x2 dimension');
    zlabel('rosenbrock value');
elseif (d > 2)
    %integration limits
    min = -2.048;
    max = 2.048;
    regn = zeros(1, 2*d, 'double');
    for i = 1:d
        regn(i) = min;
        regn(i+d) = max;
    end
    nSamplingPts1 = nSamplingPts1*(d-2);
    nSamplingPts2 = nSamplingPts2*(d-2);
end

fxn = @(x)fxnL(x); % handle for integrand function

nEval = 10; %number of integral evaluations to perform

% array to store result integral values with their standard deviation
results = zeros(2,nEval);

for i = 1:nEval
    % Run VEGAS with init = 0 to find optimal grid

    %number of performed searches for grid
```

```
    nGrid = 0;

    %flag whether a reliable grid was found
    relGrid = 0;

    while(relGrid == 0 && nGrid < 5)
        nGrid = nGrid + 1;
        tic
        [tgral0, sd0, chi2a0] = vegas(regn, fxn, 0, nSamplingPts1,...
                                      5, 0, 1, 2, 1);
        toc
        if(chi2a0 >= 0 && chi2a0 <= 4)
            relGrid = 1;
            tic
            [tgral, sd, chi2a] = vegas(regn, fxn, 1, nSamplingPts2,...
                                       1, 0, 1, 2, 1);
            toc
            results(1,i) = tgral;
            results(2,i) = sd;
        else
            fprintf('[\bCalculated grid is not reliable (chi^2/IT n = %9.2g).\n]
                chi2a0);
        end
    end %while(no reliable grid found, nGrid < 5)
    if(relGrid == 1 && nGrid ==1)
        fprintf('First grid reliable\n');
    elseif(relGrid == 1)
        fprintf('%d. grid reliable\n', nGrid);
    elseif(relGrid == 0 && nGrid == 5)
        fprintf('[\bNo reliable grid found.\n]\b');
    end
end %for(evaluations)

%true value of integral
result = 0;
if(d == 2)
    result = 28692225;
elseif(d == 5)
    result = 2278412.496;
end
figure('Name','Results', 'NumberTitle', 'off');
line([0 nEval+1], [result result], 'Color', 'g'); hold on;
xlabel('Integration'); ylabel('Integration result');
errorbar(1:nEval, results(1,:), results(2,:),...
    'LineStyle',':','Color','b', 'Marker','x', 'MarkerEdgeColor', 'r');
end

% Computes integrand for vegas algorithm
function fxnValue = fxnL(x)
%
%   x   - sample point in integration volume
%
```

```
d = numel(x); %dimension of rosenbrock function

fxnValue = 0;
for i = 1:d-1
    fxnValue = fxnValue + ((1-x(i))^2 + 100*(x(i+1) - x(i)^2)^2);
end

end
```

B.3. Bestimmung der Peakanzahl eines simulierten Spektrums

B.3.1. Peakform: Lorentz-Funktion mit zwei Modellparametern (x_0, w)

```
function testLF_2mp
%
% Generates dummy spectrum and its 'measured' data with peaks in form of
% normed lorentz function with 2 model parameters.
% Runs several evaluations to determine number of peaks (N) within spectrum
% by calculating the posterior probabilities P(N | D,I).
%

%number of model parameters (delta,w)
nmp = 2;

% values for model parameters of simulated data
dmp = zeros(nmp, 5);% nmp, number of peaks(5)
dmp(1,:) = [1.6 1.84 1.3 3.5 3.15]; % x0 - positions of amplitudes
dmp(2,:) = [0.13 0.37 0.26 0.35 0.5]; % w - FWHMs

% generate simulated/ dummy data
pd = 0.0:0.01:5;
dummySpectrum = dummySpec(dmp, pd);

% Plot dummy data
figure('Name','Dummy data', 'NumberTitle', 'off');
plot(pd, dummySpectrum);
xlabel('chemical shift');
ylabel('signal intensity');

%generate measured data
mstep = 0.1; % step for measuring points along x-axis
pdm = 0.0:mstep:5;
dataValue(:,1) = pdm;
dataValue(:,2) = dummySpec(dmp, pdm);

% Plot measured data
figure('Name','Data', 'NumberTitle', 'off');
plot(dataValue(:, 1), dataValue(:, 2), '+');
xlabel('chemical shift');
ylabel('signal intensity');

% maximal number of peaks for evaluations
n_max = 7;%-1;
while(isempty(n_max) || ~isnumeric(n_max) ...
          || (n_max < 1) || (n_max ~= fix(n_max)))
   n_max = input('Maximum number of peaks for estimates: ');

   if(isempty(n_max) || ~isnumeric(n_max) ...
          || (n_max < 1) || (n_max ~= fix(n_max)))
       fprintf(2, 'Number of peaks should be a positive integer.\n');
   end
```

```
end

%integration limits for positions of amplitudes
x0_min = 1;
x0_max = 4;

%integration limits for FWHMs
w_min = 0.1;
w_max = 0.8;

%array of integration limits for one peak
ranges = [x0_min w_min x0_max w_max];

% array for natural logaritm values of calculated probability P(N|D,I)
logVec = zeros(1, n_max, 'double');
minLogProb = 0;
maxLogProb = 0;

% number of evaluations to perform to find max P(N | D,I)
nEval = 10;%0;
while(isempty(nEval) || ~isnumeric(nEval) ||...
        (nEval < 1) || (nEval ~= fix(nEval)))
    title = 'Number of evaluations for P(N | D,I)';
    nEval = input([title ': ']);

    if(isempty(nEval) || ~isnumeric(nEval) ||...
        (nEval < 1) || (nEval ~= fix(nEval)))
      fprintf(2, [title ' should be a positive integer.\n']);
    end
end %while(input of evaluations' number)

% array for amounts of max probability that N achieved
results = zeros(1,n_max);

for nIt = 1:nEval
    fprintf('\n\n%d. Evaluation of dummy data with 2 model parameters\n', nIt);

    %number of peaks with maximal posibility in data
    maxProbN = 0;

    % flag whether an integral was left out because no reliable grid
    % was found for it (before an integral is left out of results,
    % try 5 times to find an optimal, reliable grid)
    leftoutIntegral = 0;

    % Calculates P(N|D,I) for N = 1 to n_max to find max probability
    for n = 1:n_max

        % calculate initial number of sample points
        % by precision of integral parameters
        nsampl = (x0_max - x0_min)/mstep *n + (w_max - w_min)/mstep *n;
```

```
% number of sample points for first call to vegas
% to find optimal grid
nSamplingPts1 = nsampl;

% number of sample points for second call to vegas
% to calculate integral
nSamplingPts2 = nsampl*10;

% Create function handle to integrand function
fxn = @(x)fxnL(x, n, nmp, dataValue);

if(n == 1)
    regn = ranges;
else
    % Set range vector (regn) of volume for integration
    regn = zeros(1, 2*n*nmp, 'double'); %length= 2*dim, dim= n*nmp
    for i = 1:n
        for mp = 1:nmp
            %set lower limit(s)
            regn(mp + nmp*(i-1)) = ranges(mp);

            %set upper limit(s)
            regn(mp + n*nmp + nmp*(i-1)) = ranges(mp+nmp);
        end %for number of model parameters
    end %for number of peaks
end

% Run VEGAS with init = 0 to find optimal grid

%number of performed searches for grid
nGrid = 0;

%flag whether a reliable grid was found
relGrid = 0;

while(relGrid == 0 && nGrid < 5)
    nGrid = nGrid + 1;
    tic
    init = int32(0);

    %number of iterations
    nit = int32(5);

    [tgral0, sd0, chi2a0] = vegas(regn, fxn, init, ...
                    int32(nSamplingPts1), nit, int32(0), 0, nmp, 0);
    toc

    % Run VEGAS with init = 1 if found grid is reliable
    % Calculate integral with found grid
    if(chi2a0 >= 0 && chi2a0 <= (nit-1))
        relGrid = 1;
        tic
```

```
            init = int32(1);

            %number of iterations
            nit = int32(1);

            [tgral, sd, chi2a] = vegas(regn, fxn, init,...
                        int32(nSamplingPts2), nit, int32(0), 0, nmp, 0);
            toc

            %Calculate P(N| D,I)
            div = ((x0_max - x0_min)*(w_max - w_min))^double(n);
            postN = (double(factorial(n))/ div) * tgral;
            fprintf('P(%d | D, I) = %10.7g\n', n, postN);

            logVec(n) = log(postN);

            % find min and max log(postN) for results' plot y-axis
            if(n == 1)
                minLogProb = logVec(n);
                maxLogProb = logVec(n);
                maxProbN = 1;
            elseif(logVec(n) < minLogProb)
                minLogProb = logVec(n);
            elseif(logVec(n) > maxLogProb)
                maxLogProb = logVec(n);
                maxProbN = n;
            end
        else
            title = '[\bCalculated grid for ';
            title2 = '.peak is not reliable (chi^2/IT n = ';
            fprintf([title '%2d' title2 '%9.2g).\n]\b'], n, chi2a0);
        end
    end %while(optimal grid not found, nGrid < 5)
    if(relGrid == 1 && nGrid ==1)
        fprintf('First grid reliable\n');
    elseif(relGrid == 1)
        fprintf('%d. grid reliable\n', nGrid);
    elseif(relGrid == 0 && nGrid == 5)
        title = '[\bNo reliable grid for ';
        title2 = '.peak found, will be left out in results.';
        fprintf([title '%2d' title2 '\n]\b'], n);
        leftoutIntegral = 1;
    end
end %for(1 to n_max peaks)
if(leftoutIntegral)
    title = 'At least one calculated integral was not reliable';
    msgbox([title ' and left out in results'], 'Warning','warn');
end

%plot P(N| D, I)
figure('Name','Results: Posterior probabilities for the number of peaks',...
        'NumberTitle','off');
```

```
    plot(1:n_max, logVec, '-o');
    axis([0 n_max+1 minLogProb-0.5 maxLogProb+0.5]);
    xlabel('number of peaks (N)');
    ylabel('log(P(N| D, I))');

    results(maxProbN) = results(maxProbN)+1;
end %for(evaluations)

%plot results
figure('Name','Results: number of peaks with maximal P(N| D, I)',...
       'NumberTitle','off');
plot(1:n_max, results, '-o');
axis([0 n_max+1 0 nEval+1]);
xlabel('number of peaks');
ylabel('number of maximal P(N| D, I) calculated');
end

% Computes integrand for vegas algorithm
function fxnValue = fxnL(x, n, nmp, dataValue)
%
%   x   - sample point in integration volume
%         array of random values for positions of amplitudes and FWHMs
%          for each of n peaks
%   n   - number of peaks
%   nmp - number of model parameters
%   dataValue - matrix of measured data values
%

%number of measuring points
m = int32(size(dataValue, 1));

chi2 = 0;
for k = 1:m
    %sum of lorentz functions(1 to N)
    sLf = 0;
    for j = 1:n
       iDeltaValue = 1 + nmp*(j-1);
       sLf = sLf + lorentzF(x, iDeltaValue, dataValue(k, 1));
    end %for(number of peaks)
    chi2 = chi2 + ((sLf - dataValue(k, 2))^2);
end %for(measuring points)
fxnValue = exp(-(1/2)*chi2);

end

% Calculates value of lorentz function with given parameters
function lfValue = lorentzF(x, iDeltaValue, p)
%
%   x - vector of parameter values for position of amplitude and FWHM
%   iDeltaValue - index of amplitude position value within x
%                 for currently requested peak
%   p - x /control variable values of measuring points
```

```
%
% lorentz funcion := 1/pi * (w_j/2)/((delta(k) - delta_j)^2 + (w_j/2)^2))
%
% model params: x0, FWHM(w)

div = (p - x(iDeltaValue))^double(2) + (x(iDeltaValue+1)/double(2))^double(2);

if(div ~= 0)
    s = (x(iDeltaValue+1)/double(2)) /div;
    lfValue = (1.0/pi) * s;
else
    lfValue = 0;
    fprintf('Value of lorentz function is 0 because of zero divisor.');
end

end

% Calculates dummy spectrum
function spec = dummySpec(dmp, p)
%
% dmp - data of model parameters for dummy spectrum
% p - control variable, x-axis measure points
%

%number of model parameters
nmp = 2; % (x0, FWHM)

%number of peaks
n = numel(dmp)/nmp;
spec = 0;
for j=1:n

    %normalised lorentz formula without A
    spec = spec + (1/pi * (dmp(2, j)/2.0)./...
        ((p-dmp(1, j)).^2 + (dmp(2, j)/2.0)^double(2)));
end

end
```

B.3.2. Peakform: Lorentz-Funktion mit drei Modellparametern (x_0, w, A)

```
function dummyLf_3mp
%
% Generates dummy spectrum and its 'measured' data with peaks in form of
% lorentz function with 3 model parameters.
% Runs several evaluations to determine number of peaks (N) within spectrum
% by calculating the posterior probabilities P(N | D,I).
%

%number of model parameters (x0, w, a)
nmp = 3;

% values for model parameters of simulated data
xd = 0:0.01:4;
dmp = zeros(nmp, 2);% nmp, number of peaks(2)
dmp(1,:) = [1.3  2.5]; % x0 - positions of amplitudes
dmp(2,:) = [0.25  0.35]; %w - FWHMs
dmp(3,:) = [2  2.3]; % a - amplitudes
dummySpectrum = dummySpec(dmp, xd);
figure('Name','Dummy data', 'NumberTitle', 'off');
plot(xd, dummySpectrum);
xlabel('chemical shift');
ylabel('signal intensity');

% simulate measured data
mstep = 0.1; % step for measuring points along x-axis
xdm = 0:mstep:4;
mSpectrum = dummySpec(dmp, xdm);
dataValue = zeros(numel(xdm), 2, 'double');
dataValue(:, 1) = xdm;
dataValue(:, 2) = mSpectrum;
figure('Name','Data', 'NumberTitle', 'off');
plot(xdm, mSpectrum, '+');
xlabel('chemical shift');
ylabel('signal intensity');

% maximal number of peaks to calculate P(N| D, I) for
n_max = 7;%-1;
while(isempty(n_max) || ~isnumeric(n_max) ...
          || (n_max < 1) || (n_max ~= fix(n_max)))
   n_max = input('Maximum number of peaks to use for estimates: ');

   if(isempty(n_max) || ~isnumeric(n_max) ...
          || (n_max < 1) || (n_max ~= fix(n_max)))
       fprintf(2, 'Number of peaks should be a positive integer.\n');
   end
end %while(input for n_max)

%integration limits for amplitudes
a_min = 0.5;
a_max = 2.5;
```

```
%integration limits for positions of amplitudes
x0_min = 0.5;
x0_max = 3.5;

%integration limits for FWHMs
w_min = 0.1;
w_max = 0.5;

%array of integration limits for one peak
ranges = [x0_min w_min a_min x0_max w_max a_max];

%vector for natural logarithm values of calculated probability P(n|D,I)
logVec = zeros(1, n_max, 'double');
minLogProb = 0;
maxLogProb = 0;

% number of evaluations to perform to find max P(N | D,I)
nEval = 10;%0;
while(isempty(nEval) || ~isnumeric(nEval) ||...
        (nEval < 1) || (nEval ~= fix(nEval)))
    title = 'Number of evaluations for P(N | D,I)';
    nEval = input([title ': ']);

    if(isempty(nEval) || ~isnumeric(nEval) ||...
        (nEval < 1) || (nEval ~= fix(nEval)))
      fprintf(2, [title ' should be a positive integer.\n']);
    end
end %while(input of evaluations' number)

% array for amounts of max probability that N achieved
results = zeros(1,n_max);

for nIt = 1:nEval
    fprintf('\n\n%d. Evaluation of dummy data with 3 model parameters\n', nIt);

    %number of peaks with maximal posibility in data
    maxProbN = 0;

    % flag whether an integral was left out because no reliable grid
    % was found for it (before an integral is left out of results,
    % try 5 times to find an optimal, reliable grid)
    leftoutIntegral = 0;

    % Calculates P(N|D,I) for N = 1 to n_max to find max probability
    for n = 1:n_max

        % calculate initial number of sample points
        % by precision of integral parameters
        nsampl = (x0_max - x0_min)/mstep *n + (w_max - w_min)/mstep *n ...
                    + (a_max - a_min)/mstep *n;
```

```
% number of sample points for first call to vegas
% to find optimal grid
nSamplingPts1 = nsampl;

% number of sample points for second call to vegas
% to calculate integral
nSamplingPts2 = nsampl*10;

% Create function handle to function to be integrated
fxn = @(x)fxnL(x, n, nmp, dataValue);

% Set range vector (regn) of volume for integration
regn = zeros(1, 2*n*nmp, 'double'); %length= 2*dim, dim= n*nmp
for d = 1:n
    for mp = 1:nmp
        %set lower limit(s)
        regn(mp + nmp*(d-1)) = ranges(mp);

        %set upper limit(s)
        regn(mp + n*nmp + nmp*(d-1)) = ranges(mp+nmp);
    end
end

% Run VEGAS with init = 0 to find optimal grid

%number of run searches for grid
nGrid = 0;

%flag whether a reliable grid was found
relGrid = 0;

% try 5 times to find a reliable grid for integration
while(relGrid == 0 && nGrid < 5)
    nGrid = nGrid +1;
    tic
    init = int32(0);

    %number of iterations
    nit = int32(5);

    [tgral0, sd0, chi2a0] = vegas(regn, fxn, init, ...
                    int32(nSamplingPts1), nit, int32(0), 0, nmp, 0);
    toc

    % Run VEGAS with init = 1 if found grid is reliable
    % Calculate integral with found grid
    if(chi2a0 >= 0 && chi2a0 <= (nit-1))
        relGrid = 1;
        tic
        init = int32(1);

        %number of iterations
```

```
            nit = int32(1);

            [tgral, sd, chi2a] = vegas(regn, fxn, init,...
                        int32(nSamplingPts2), nit, int32(0), 0, nmp, 0);
            toc

            %Calculate P(N | D,I)
            div = ((a_max-a_min)*(x0_max-x0_min)*(w_max-w_min))^double(n);
            postN = (double(factorial(n))/ div) * tgral;
            fprintf('P(%d | D, I) = %10.7g\n', n, postN);

            logVec(n) = log(postN);

            % find minimum log(postN) for results' plot axis
            if(logVec(n) ~= 0)
                if(n == 1)
                    minLogProb = logVec(n);
                    maxLogProb = logVec(n);
                    maxProbN = 1;
                elseif(logVec(n) < minLogProb)
                    minLogProb = logVec(n);
                elseif(logVec(n) > maxLogProb)
                    maxLogProb = logVec(n);
                    maxProbN = n;
                end
            end
        else
            title = '[\bCalculated grid for ';
            title2 = '.peak is not reliable (chi^2/IT n =  ';
            fprintf([title '%2d' title2 '%9.2g).\n]\b'], n, chi2a0);
        end
    end %while(optimal grid not found, nGrid < 5)
    if(relGrid == 1)
        fprintf('%d. grid reliable\n', nGrid);
    elseif(relGrid == 0 && nGrid == 5)
        title = '[\bNo reliable grid for ';
        title2 = '.peak found, will be left out in results.';
        fprintf([title '%2d' title2 '\n]\b'], n);
        leftoutIntegral = 1;
    end
end %for(1 to n_max peaks)
if(leftoutIntegral)
    title = 'At least for one number of peaks there was no reliable';
    msgbox([title 'and left out in results'], 'Warning','warn');
end

%plot P(N| D, I)
figure('Name','Results: Posterior probabilities for the number of peaks',...
       'NumberTitle','off');
plot(1:n_max, logVec, '-o');
axis([0 n_max+1 minLogProb-0.5 maxLogProb+0.5]);
xlabel('number of peaks (N)');
```

```
    ylabel('log(P(N|␣D,I))');

    results(maxProbN) = results(maxProbN)+1;
end %for(number of evaluations)

%plot results
figure('Name','Results:␣number␣of␣peaks␣with␣maximal␣P(N|␣D,␣I)',...
        'NumberTitle','off');
plot(1:n_max, results, '-o');
axis([0 n_max+1 0 nEval+1]);
xlabel('number␣of␣peaks');
ylabel('number␣of␣maximal␣P(N|␣D,␣I)␣calculated');

end

% Computes integrand for vegas algorithm
function fxnValue = fxnL(x, n, nmp, dataValue)
%
%   x   - sample point in integration volume
%         array of random values within integral limits for
%         positions of amplitudes, FWHMs and amplitudes for each of n peaks
%   n   - number of peaks
%   nmp - number of model parameters
%   dataValue - matrix of measured data values
%

%number of measurements
m = int32(size(dataValue, 1));

chi2 = 0;
for k = 1:m
    %sum of lorentz functions(1 to N)
    sf = 0;
    for j = 1:n
       % index of delta value within 'x'-array
       iDeltaValue = 1 + nmp*(j-1);
       sf = sf + lorentzF(x, iDeltaValue, dataValue(k, 1));
    end
    chi2 = chi2 + ((sf - dataValue(k, 2))^double(2));
end
fxnValue = exp(-(1/2)*chi2);

end

% Calculates value of lorentz function with given parameters
function lfValue = lorentzF(x, iDeltaValue, p)
%
%   x - vector of parameter values for amplitude,
%       position of amplitude and FWHM
%   iDeltaValue - index of amplitude position value within x
%                 for currently requested peak
%   p - control variable value (x-axis value to calculate
```

```
%       lorentz function value for)
%
% model params: position of amplitude, FWHM, amplitude
    lfValue = x(iDeltaValue+2)/...
                (1+((p-x(iDeltaValue))/(x(iDeltaValue+1)/2.0))^double(2));

end

% Calculates dummy spectrum
function spec = dummySpec(dmp, p)
%
% dmp - data of model parameters
% p - control variable, x-axis measure points
%

%number of model parameters
nmp = 3; % (x0, FWHM, a)

%number of peaks
n = numel(dmp)/nmp;
spec = 0;
for j=1:n
    spec = spec + dmp(3, j) ./(1.0+((p-dmp(1, j))/ (dmp(2, j)/2.0)).^double(2));
end

end
```

B.4. Bestimmung der Peakanzahl mit Messdaten

B.4.1. Anwendung mit der VEGAS-Integration

```
function nmr_pvpVegas_x0pwpA

%number of model parameters (delta,w, a)
nmp = 3;

dataValue = load('E:\Chrissi\Diplomarbeit\Daten\13C-FID_27MA17-BT.010.mat');
m = numel(dataValue.E);
dataValues = zeros(m, 2);
dataValues(:,1) = dataValue.E(1:m);
dataValues(:,2) = dataValue.sp1(1:m);

%precision of data along x-axis
mstep = dataValues(2,1) - dataValues(1,1);

% norm data values by their sum
sumDV = sum(dataValues(:,2));
dataValues(:, 2) = dataValues(:, 2)./ sumDV;

% calibrated eta value
etac = -1;
% etaInput = input('Do you have a value for the proportion of lorentz %s',...
%           'function within the pseudo voigt profile, eta, yet?(y/n): ', 's');
% if(strcmp(etaInput, 'y'))
%     while(isempty(etac) || ~isnumeric(etac) || (etac < 0) || (etac > 1))
%         etac = input('Proportion of lorentz function within %s',...
%                      'pseudo voigt profile, eta: ');
%
%         if(isempty(etac) || ~isnumeric(etac) || (etac < 0) || (etac > 1))
%             fprintf(2, 'Eta should be a number between 0 and 1.\n');
%         end
%     end
% elseif(strcmp(etaInput, 'n'))
%     fprintf('Eta will be estimated during evaluation.\n');
% end

% Plot measured data
figure('Name','Data', 'NumberTitle', 'off');
plot(dataValues(:, 1), dataValues(:, 2), '+');
xlabel('chemical shift');
ylabel('signal intensity');

evalFin = 0;
while(evalFin == 0)
    evalGo = 0;
    while(evalGo == 0)
        x_start = 136.555;%-Inf;
        while(isempty(x_start) || ~isnumeric(x_start) || (x_start == -Inf)...
            || (x_start < dataValues(1,1)) || (x_start > dataValues(m,1)))
```

```
            title = 'Lower x-value for evaluation interval';
            x_start = input([title ': ']);

            if(isempty(x_start) || ~isnumeric(x_start) || (x_start == -Inf)...
             || (x_start < dataValues(1,1)) || (x_start > dataValues(m,1)))
                condition = 'positive number within the measuring points';
                fprintf(2, [title ' should be a' condition '.\n']);
            end
        end

        x_end = 136.59;%Inf;
        while(isempty(x_end) || ~isnumeric(x_end) || (x_start == -Inf)...
             || (x_start < dataValues(1,1)) || (x_start > dataValues(m,1)))
             title = 'Upper x-value for evaluation interval';
            x_start = input([title ': ']);

            if(isempty(x_end) || ~isnumeric(x_end) || (x_start == -Inf)...
             || (x_start < dataValues(1,1)) || (x_start > dataValues(m,1)))
                condition = 'positive number within the measuring points';
                fprintf(2, [title ' should be a' condition '.\n']);
            end
        end
        iStart = find(dataValues(:, 1) >= x_start, 1);
        iEnd = find(dataValues(:, 1) >= x_end, 1);
        figure('Name','Evaluation interval of Data', 'NumberTitle', 'off');
        plot(dataValues(iStart:iEnd, 1), dataValues(iStart:iEnd, 2), '+');
        xlabel('chemical shift');
        ylabel('signal intensity');

        ok = input('Evaluation interval good to go?(y/n): ', 's');
        evalGo = strcmp(ok, 'y');
    end %while(input for evaluation interval)

    dataVEval = zeros((iEnd-iStart)+1, 2);
    dataVEval(:, 1) = dataValues(iStart:iEnd, 1);
    dataVEval(:, 2) = dataValues(iStart:iEnd, 2);

    n_max = 7;%-1;
    while(isempty(n_max) || ~isnumeric(n_max) ...
            || (n_max < 0) || (n_max ~= fix(n_max)))
        title = 'Maximal number of peaks for estimation';
        n_max = input([title ': ']);

        if(isempty(n_max) || ~isnumeric(n_max) ...
            || (n_max < 0) || (n_max ~= fix(n_max)))
            fprintf(2, [title ' should be a positive integer.\n']);
        end
    end %while(input n_max)

    eX0 = 1;
    %input('Previous estimation of positions of amplitudes done?(y/n): ', 's');
%     eX0 = strcmp(eX0, 'y');
```

```
%     nX0 = -1;
    if(eX0 == 1)
        nX0 = 3;
        %remain order of x0 values because eta estimation relies on it
        x0 = [136.573 136.568 136.579];
%         while(isempty(nX0) || ~isnumeric(nX0) ...
%                 || (nX0 <= 0) || (nX0 ~= fix(nX0)))
%             title = 'Number of estimated positions of amplitudes';
%             nX0 = input([title ': ']);
%
%             if(isempty(nX0) || ~isnumeric(nX0) ...
%                 || (nX0 <= 0) || (nX0 ~= fix(nX0)))
%                 fprintf(2, [title ' should be a positive integer.\n']);
%             end
%         end
%         x0 = zeros(1, nX0);
%         for iX0 = 1:nX0
%             x0(iX0) = Inf;
%             while(isempty(x0(iX0)) || ~isnumeric(x0(iX0)) ...
%                     || x0(iX0) < x_start || x0(iX0) > x_end || x0(iX0) == Inf)
%                 title = [num2str(iX0) '. estimated value of position of ];
%                 x0(iX0) = input([title 'amplitude: ']);
%
%                 if(isempty(x0(iX0)) || ~isnumeric(x0(iX0)) ...
%                     || x0(iX0) < x_start || x0(iX0) > x_end || x0(iX0) == Inf)
%                     title = 'Estimated position of amplitude should';
%                     condition = ' and lie within the evaluation interval';
%                     fprintf(2, [title 'be a number' condition '.\n']);
%                 end
%             end
%         end
    end

    a_min = 0.005;%-1;
    while(isempty(a_min) || ~isnumeric(a_min) || (a_min < 0))
        title = 'Minimum value for possible amplitudes';
        a_min = input([title ': ']);

        if(isempty(a_min) || ~isnumeric(a_min) || (a_min < 0))
            fprintf(2, [title ' should be a positive number.\n']);
        end
    end
    a_max = 0.06;%Inf;
    while(isempty(a_max) || ~isnumeric(a_max) ...
            || (a_max == Inf) || (a_max < 0))
        title = 'Maximum value for possible amplitudes';
        a_max = input([title ': ']);

        if(isempty(a_max) || ~isnumeric(a_max) ...
                || (a_max == Inf) || (a_max < 0))
            fprintf(2, [title ' should be a positive number.\n']);
        end
```

```
end

a_step = 0.01;%-1; 1.peak:0.005, 2.peak:0.01, 3.peak:0.001
while(isempty(a_step) || ~isnumeric(a_step) || (a_step <= 0))
    title = 'Step width/ precision for amplitude estimation';
    a_step = input([title ': ']);

    if(isempty(a_step) || ~isnumeric(a_step) || (a_step <= 0))
       fprintf(2, [title ' should be a positive number.\n']);
    end
end

w_min = mstep;%0;
while(isempty(w_min) || ~isnumeric(w_min) || (w_min <= 0))
    title = 'Minimum value for possible FWHMs';
    w_min = input([title ': ']);

    if(isempty(w_min) || ~isnumeric(w_min) || (w_min <= 0))
       fprintf(2, [title ' should be a positive number.\n']);
    end
end
w_max = 0.013;%5;
while(isempty(w_max) || ~isnumeric(w_max) ...
    || (w_max >= 5) || (w_max < 0))
    title = 'Maximum value for possible FWHMs';
    w_max = input([title ': ']);

    if(isempty(w_max) || ~isnumeric(w_max) ...
        || (w_max >= 5) || (w_max < 0))
        fprintf(2, [title ' should be a positive number and < 5.\n']);
    end
end

w_step = mstep;%-1;
while(isempty(w_step) || ~isnumeric(w_step) ...
        || (w_step < mstep) || (w_step <= 0))
    ms = num2str(mstep);
    title = 'Step width/ precision for FWHM estimation';
    w_step = input([title '(measuring precision: ' ms '): ']);

    if(isempty(w_step) || ~isnumeric(w_step) ...
        || (w_step < mstep) || (w_step <= 0))
        fprintf(2, [title ' should be a positive number and >= ' ms '.\n']);
    end
end

%array with integration limits for FWHM and amplitude of one peak
ranges = [w_min a_min w_max a_max];

x0_min = 0; x0_max = 0;

%(n-nX0) positions of amplitudes are to be estimated/ marginalised
```

```
if(nX0 < n_max)
    x0_min = 136.56;%-Inf;
    while(isempty(x0_min) || ~isnumeric(x0_min) || (x0_min == -Inf))
        title = 'Minimum value for possible positions of amplitudes';
        x0_min = input([title ': ']);

        if(isempty(x0_min) || ~isnumeric(x0_min) || (x0_min == -Inf))
            fprintf(2, [title ' should be a number.\n']);
        end
    end
    x0_max = 136.585;%Inf;
    while(isempty(x0_max) || ~isnumeric(x0_max) ...
            || (x0_max < x0_min) || (x0_max == Inf))
        title = 'Maximum value for possible positions of amplitudes';
        x0_max = input([title ': ']);

        if(isempty(x0_max) || ~isnumeric(x0_max) ...
            || (x0_max < x0_min) || (x0_max == Inf))
           fprintf(2, [title ' should be a number.\n']);
        end
    end
    ranges = [x0_min w_min a_min x0_max w_max a_max];
end

%vector for natural logaritm values of calculated probability P(n|D,I)
logVec = zeros(1, n_max, 'double');
minLogProb = 0;
maxLogProb = 0;

nEval = 10;%-1;
while(isempty(nEval) || ~isnumeric(nEval) ||...
        (nEval < 1) || (nEval ~= fix(nEval)))
    title = 'Number of calculations of integral';
    nEval = input([title ': ']);

    if(isempty(nEval) || ~isnumeric(nEval) ||...
        (nEval < 1) || (nEval ~= fix(nEval)))
      fprintf(2, [title ' should be a positive integer.\n']);
    end
end %while(input of evaluations' number)

% array for amounts of max probability that N achieved
results = zeros(1,n_max);

for nIt = 1:nEval
    title = 'Evaluation of measured data (given positions';
    fprintf(['\n\n%d. ' title 'of amplitude)\n'], nIt);

    %number of peaks with maximal posibility in data
    maxProbN = 0;

    % flag whether an integral was left out because no reliable grid
```

```
% was found for it (before an integral is left out of results,
% try 5 times to find an optimal, reliable grid)
leftoutIntegral = 0;

% Perform 1 to n_max calculations for P(N|D,I)
% to find max probability
for n = 1:n_max
    % no calibrated value for eta
    if(etac == -1)
        %
        if(n==1)
            eta = 1;
        elseif(n==2)
            eta = 1;
        elseif(n==3)
            eta = 0.5;
        elseif(n == 4)
            eta = 0.58;
        elseif(n == 5)
            eta = 0.62;
        elseif(n == 6)
            eta = 0.53;
        elseif(n == 7)
            eta = 0.6;
        else
            eta = -1;
            cftool(dataVEval(:,1), dataVEval(:,2))
            pause;
        end
        while(isempty(eta) || ~isnumeric(eta) ...
            || (eta < 0) || (eta > 1))
           eta = input('By fitting tool calculated value for eta: ');

           if(isempty(eta) || ~isnumeric(eta) ...
               || (eta < 0) || (eta > 1))
               fprintf(2, 'Eta should be a number between 0 and 1.\n');
           end
        end
    else
        eta = etac;
    end

    if(nX0 < n)

        % calculate initial number of sample points
        % by precision of integral parameters
        nsampl = (x0_max - x0_min)/mstep *(n-nX0) ...
            + (w_max - w_min)/mstep *n + (a_max - a_min)/a_step *n;

        % Set range vector (regn) of volume for integration
        regn = zeros(1, 2*n*(nmp-1) + 2*(n-nX0), 'double');
        for i = 1:(n-nX0)
```

```
            for mp = 1:nmp
            %set lower limit
            regn(mp + nmp*(i-1)) = ranges(mp);

            %set upper limit
            regn(mp+(n-nX0)*nmp+nX0*(nmp-1)+nmp*(i-1))=ranges(mp+nmp);
            end
        end %for(peaks with unknown values for all model parameters)
        for i = 1:nX0
            for mp = 1:(nmp-1)
        %set lower limit
        regn(mp + (n-nX0)*nmp + (nmp-1)*(i-1)) = ranges(mp+1);

        %set upper limit
        upperLimit = ranges(mp+1+nmp);
        regn(mp+2*(n-nX0)*nmp+nX0*(nmp-1)+(nmp-1)*(i-1))=upperLimit;
            end
        end %for(peaks with unknown values for two model parameters(w,A)
    else

        % calculate initial number of sample points
        % by precision of integral parameters
        nsampl = (w_max - w_min)/mstep *n + (a_max - a_min)/a_step *n;

        % Set range vector (regn) of volume for integration
        %length= 2*dim, dim= n*(nmp-1)
        regn = zeros(1, 2*n*(nmp-1), 'double');
        x0r = 0;
        if(numel(ranges) > 2*(nmp-1))
            x0r = 1;
        end
        for i = 1:n
            for mp = 1:(nmp-1)
                %set lower limit
                regn(mp + (nmp-1)*(i-1)) = ranges(mp+x0r);

                %set upper limit
                regn(mp+n*(nmp-1)+(nmp-1)*(i-1))=ranges(mp+x0r+nmp);
            end %for(model parameters w and A)
        end %for(peaks)
    end
    % number of sample points for first call to vegas
    % to find optimal grid
    nSamplingPts1 = nsampl;

    % number of sample points for second call to vegas
    % to calculate integral
    nSamplingPts2 = nsampl*10;

    % Create function handle to function to be integrated
    fxn = @(x)fxnL(x, n, nmp, x0, eta, dataVEval);
```

```
% Run VEGAS with init = 0 to find optimal grid

%number of run searches for grid
nGrid = 0;

%flag whether a reliable grid was found
relGrid = 0;

while(relGrid == 0 && nGrid < 5)
    nGrid = nGrid +1;
    tic
    init = int32(0);

    %number of iterations
    nit = int32(5);

    [tgral0, sd0, chi2a0] = vegas(regn, fxn, init, ...
              int32(nSamplingPts1), nit, int32(0), 0, nmp, 0);
    toc

    % Run VEGAS with init = 1 if found grid is reliable
    % Calculate integral with found grid
    if(chi2a0 >= 0 && chi2a0 <= (nit-1))
        relGrid = 1;
        tic
        init = int32(1);

        %number of iterations
        nit = int32(1);

        [tgral, sd, chi2a] = vegas(regn, fxn, init, ...
              int32(nSamplingPts2), nit, int32(0), 0, nmp, 0);
        toc

        %Calculate P(N | D,I)

        % prior for positions of amplitudes
        x0pr = 1;
        if(nX0 < n)
            x0pr = (x0_max - x0_min)^double(n-nX0);
        end
        div = ((a_max - a_min)*(w_max - w_min))^double(n) * x0pr;
        postN = (double(factorial(n))/ div) * tgral*sumDV;
        fprintf('P(%d | D, I) = %10.7g\n', n, postN);

        logVec(n) = log(postN);

        % find minimum log(postN) for results' plot axis
        if(n == 1)
            minLogProb = logVec(n);
            maxLogProb = logVec(n);
            maxProbN = 1;
```

```
                    elseif(logVec(n) < minLogProb)
                        minLogProb = logVec(n);
                    elseif(logVec(n) > maxLogProb)
                        maxLogProb = logVec(n);
                        maxProbN = n;
                    end
                else
                    title = '[\bCalculated grid for ';
                    title2 = '.peak is not reliable (chi^2/IT n =';
                    fprintf([title '%2d' title2 '%9.2g).\n]\b'], n, chi2a0);
                end
            end %while(no reliable grid found and nGrid < 5)
            if(relGrid == 1 && nGrid ==1)
                fprintf('First grid reliable\n');
            elseif(relGrid == 1)
                fprintf('%d. grid reliable\n', nGrid);
            elseif(relGrid == 0 && nGrid == 5)
                title = '[\bNo reliable grid for ';
                title2 = '.peak found, will be left out in results.';
                fprintf([title '%2d' title2 '\n]\b'], n);
                leftoutIntegral = 1;
            end
        end %for(1 to n_max)
        if(leftoutIntegral)
            title = 'At least one calculated integral was not reliable';
            msgbox([title ' and left out in results'], 'Warning','warn');
        end

        %plot results
        figure('Name','Results: Posterior probabilities for the number of peaks'
                'NumberTitle','off');
        plot(1:1:n_max, logVec, '-o');
        axis([0 n_max+0.5 minLogProb-0.5 maxLogProb+0.5]);
        xlabel('number of peaks (N)');
        ylabel('log(P(N| D,I))');

        results(maxProbN) = results(maxProbN)+1;
    end %for(evaluations)

    %plot results
    figure('Name','Results: number of peaks with maximal P(N| D, I)',...
            'NumberTitle','off');
    plot(1:n_max, results, '-o');
    axis([0 n_max+1 0 nEval+1]);
    xlabel('number of peaks');
    ylabel('number of maximal P(N| D, I) calculated');

    ok = input('Do you want to evaluate data within another interval?(y/n): ',...
                's');
    evalFin = strcmp(ok, 'n');
end %while(finish evaluation)
end
```

```
% Computes integrand for vegas algorithm
function fxnValue = fxnL(x, n, nmp, x0, eta, dataValue)
%
%   x    - array of sample points for FWHMs and amplitudes for each of n
%          peaks; if n > given positions of amplitudes, it contains sample
%          points for positions of amplitudes that are to be estimated
%   n    - number of peaks
%   nmp - number of model parameters
%   x0   - array of given positions of amplitudes
%   eta - lorentz proportion within pseudo voigt profile
%   dataValue - matrix of measured data values
%

%number of measuring points
m = int32(size(dataValue, 1));

chi2 = 0;
for k = 1:m
    %sum of pseudo voigt profiles(1 to N)
    sLf = 0;
    for j = 1:n
        sLf = sLf + psVoigtP(x, j, nmp, n, x0, eta, dataValue(k, 1));
    end
    chi2 = chi2 + ((sLf - dataValue(k, 2))^2);
end
fxnValue = exp(-(1/2)*chi2);

end


% Calculates value of pseudo voigt profile with given parameters
function pvpValue = psVoigtP(x, j, nmp, n, x0, eta, p)
%
%   x - array of parameter values for FWHM, amplitude and if needed
%       position of amplitude
%   j - peak whose function value is requested
%   nmp - number of model parameters
%   n - number of peaks for current spectrum
%   x0 - given values for position of amplitude
%   eta - lorentz proportion within pseudo voigt profile
%   p - control variable value (x axis value to calculate
%       pseudo voigt profile value for)
%
% model params: position of amplitude, FWHM, amplitude

%number of given positions of amplitude
nX0 = numel(x0);

if(nX0 < j)
    iDeltaValue = 1 + nmp*(j-nX0-1);
    x0j = x(iDeltaValue);
```

```
    w = x(iDeltaValue+1);
    a = x(iDeltaValue+2);
else % there is a given position of amplitude for requested peak
    p3mp = 0;
    if(n > nX0)
        p3mp = nmp*(n-nX0);
    end
    iWValue = 1 + p3mp + (nmp-1)*(j-1);
    x0j = x0(j);
    w = x(iWValue);
    a = x(iWValue+1);
end

gf = exp(-log(2)*((p-x0j)/(w/2.0))^double(2));
lf = 1 /(1+((p-x0j)/(w/2.0))^double(2));
pvpValue = a*(eta*lf + (1 - eta)*gf);
end
```

B.4.2. Anwendung mit numerischer Integration über die Likelihood-Matrix

```
function nmr_pvp
%
% Runs evaluations to estimate number of peaks within measured spectrum.
% Estimation by likelihood matrix for 3 and 4 peaks.
%

% load measuring data from file
dataValue = load('E:\Chrissi\Diplomarbeit\Daten\13C-FID_27MA17-BT.010.mat');
m = numel(dataValue.E);
dataValues = zeros(2, m);
dataValues(1,:) = dataValue.E(1:m);
dataValues(2,:) = dataValue.sp1(1:m);

%precision of data along x-axis
mstep = dataValues(1,2) - dataValues(1,1);

% norm data values by their sum
sumDV = sum(dataValues(2,:));
dataValues(2,:) = dataValues(2,:)./ sumDV;

% calibrated eta value
etac = -1;
% etaInput = input('Do you have a value for the proportion of lorentz %s',...
%           'function within the pseudo voigt profile, eta, yet?(y/n): ', 's');
% if(strcmp(etaInput, 'y'))
%     while(isempty(etac) || ~isnumeric(etac) || (etac < 0) || (etac > 1))
%         etac = input('Proportion of lorentz function within %s',...
%                      'pseudo voigt profile, eta: ');
%
%         if(isempty(etac) || ~isnumeric(etac) || (etac < 0) || (etac > 1))
%             fprintf(2, 'Eta should be a number between 0 and 1.\n');
%         end
%     end
% elseif(strcmp(etaInput, 'n'))
%     fprintf('Eta will be estimated during evaluation.\n');
% end

% Plot measured data
figure('Name','Data', 'NumberTitle', 'off');
plot(dataValues(1,:), dataValues(2,:), '+');
xlabel('chemical shift');
ylabel('signal intensity');

evalFin = 0;
while(evalFin == 0)
    evalGo = 0;
    while(evalGo == 0)
        x_start = 136.555;%-Inf;
        while(isempty(x_start) || ~isnumeric(x_start) || (x_start == -Inf))
            title = 'Lower x-value for evaluation interval';
```

```
        x_start = input([title ': ']);

        if(isempty(x_start) || ~isnumeric(x_start) || (x_start == -Inf))
            fprintf(2, [title ' should be a positive number.\n']);
        end
    end

    x_end = 136.59;%Inf;
    while(isempty(x_end) || ~isnumeric(x_end) || (x_end == Inf))
        title = 'Upper x-value for evaluation interval';
        x_start = input([title ': ']);

        if(isempty(x_end) || ~isnumeric(x_end) || (x_end == Inf))
            fprintf(2, [title ' should be a positive number.\n']);
        end
    end
    iStart = find(dataValues(1,:) >= x_start, 1);
    iEnd = find(dataValues(1,:) >= x_end, 1);
    figure('Name','Current evaluation interval of Data',...
            'NumberTitle', 'off');
    plot(dataValues(1, iStart:iEnd), dataValues(2, iStart:iEnd), '+');
    xlabel('chemical shift');
    ylabel('signal intensity');

    ok = input('Evaluation interval good to go?(y/n): ', 's');
    evalGo = strcmp(ok, 'y');
end

dataVEval = zeros(2, (iEnd-iStart)+1);
dataVEval(1,:) = dataValues(1, iStart:iEnd);
dataVEval(2,:) = dataValues(2, iStart:iEnd);

n_min = 3;%-1;
while(isempty(n_min) || ~isnumeric(n_min) ...
        || (n_min < 0) || (n_min ~= fix(n_min)))
    title = 'Minimal number of peaks for estimation';
    n_min = input([title ': ']);

    if(isempty(n_min) || ~isnumeric(n_min) ...
            || (n_min < 0) || (n_min ~= fix(n_min)))
        fprintf(2, [title ' should be a positive integer.\n']);
    end
end

n_max = 4;%-1;
while(isempty(n_max) || ~isnumeric(n_max) ...
        || (n_max < 0) || (n_max < n_min) || (n_max ~= fix(n_max)))
    title = 'Maximal number of peaks for estimation';
    n_max = input([title ': ']);

    if(isempty(n_max) || ~isnumeric(n_max) ...
        || (n_max < 0) || (n_max < n_min) || (n_max ~= fix(n_max)))
```

```
            fprintf(2, [title ' should be a positive integer.\n']);
        end
    end

    eX0 = 1;
    %input('Previous estimation of positions of amplitudes done?(y/n): ', 's');
%     eX0 = strcmp(eX0, 'y');
%     nX0 = -1;
    if(eX0 == 1)
        nX0 = 3;
        x0 = [136.568 136.573 136.579];
%         while(isempty(nX0) || ~isnumeric(nX0) ...
%                 || (nX0 <= 0) || (nX0 ~= fix(nX0)))
%             title = 'Number of estimated positions of amplitudes';
%             nX0 = input([title ': ']);
%
%             if(isempty(nX0) || ~isnumeric(nX0) ...
%                 || (nX0 <= 0) || (nX0 ~= fix(nX0)))
%                 fprintf(2, [title ' should be a positive integer.\n']);
%             end
%         end
%         x0 = zeros(1, nX0);
%         for iX0 = 1:nX0
%             x0(iX0) = Inf;
%             while(isempty(x0(iX0)) || ~isnumeric(x0(iX0)) ...
%                     || x0(iX0) < x_start || x0(iX0) > x_end || x0(iX0) == Inf)
%                 title = [num2str(iX0) '. estimated value of position of ];
%                 x0(iX0) = input([title 'amplitude: ']);
%
%                 if(isempty(x0(iX0)) || ~isnumeric(x0(iX0)) ...
%                     || x0(iX0) < x_start || x0(iX0) > x_end || x0(iX0) == Inf)
%                     title = 'Estimated position of amplitude should';
%                     condition = ' and lie within the evaluation interval';
%                     fprintf(2, [title 'be a number' condition '.\n']);
%                 end
%             end
%         end
    end

    a_min = 0.005;%-1;
    while(isempty(a_min) || ~isnumeric(a_min) || (a_min < 0))
        title = 'Minimum value for possible amplitudes';
        a_min = input([title ': ']);

        if(isempty(a_min) || ~isnumeric(a_min) || (a_min < 0))
            fprintf(2, [title ' should be a positive number.\n']);
        end
    end
    a_max = 0.06;%Inf;
    while(isempty(a_max) || ~isnumeric(a_max) ...
            || (a_max == Inf) || (a_max < 0))
        title = 'Maximum value for possible amplitudes';
```

```
    a_max = input([title ': ']);

    if(isempty(a_max) || ~isnumeric(a_max) ...
          || (a_max == Inf) || (a_max < 0))
       fprintf(2, [title ' should be a positive number.\n']);
    end
end

a_step = 0.01;%-1; 1.peak:0.005, 2.peak:0.01, 3.peak:0.001
while(isempty(a_step) || ~isnumeric(a_step) || (a_step <= 0))
    title = 'Step width/ precision for amplitude estimation';
    a_step = input([title ': ']);

    if(isempty(a_step) || ~isnumeric(a_step) || (a_step <= 0))
       fprintf(2, [title ' should be a positive number.\n']);
    end
end
pALimits = [a_min a_step a_max];

w_min = mstep;%0;
while(isempty(w_min) || ~isnumeric(w_min) || (w_min <= 0))
    title = 'Minimum value for possible FWHMs';
    w_min = input([title ': ']);

    if(isempty(w_min) || ~isnumeric(w_min) || (w_min <= 0))
       fprintf(2, [title 'should be a positive number.\n']);
    end
end
w_max = 0.013;%5;
while(isempty(w_max) || ~isnumeric(w_max) || (w_max >= 5) || (w_max < 0))
    title = 'Maximum value for possible FWHMs';
    w_max = input([title ': ']);

    if(isempty(w_max) || ~isnumeric(w_max) || (w_max >= 5) || (w_max < 0))
       fprintf(2, [title ' should be a positive number, smaller than 5.\n'])
    end
end

w_step = mstep;%-1;
while(isempty(w_step) || ~isnumeric(w_step) ...
        || (w_step < mstep) || (w_step <= 0))
   ms = num2str(mstep);
   title = 'Step width/ precision for FWHM estimation';
   w_step = input([title '(measuring precision: ' ms '): ']);

   if(isempty(w_step) || ~isnumeric(w_step) ...
        || (w_step < mstep) || (w_step <= 0))
       fprintf(2, [title ' should be a positive number and >= ' ms '.\n']);
   end
end
pwLimits = [w_min w_step w_max];
```

```
    %(n-nX0) positions of amplitudes are to be estimated
    if(nX0 < n_max)
        x0_min = 136.56;%-Inf;
        title = 'Minimum value for possible positions of amplitudes';
        while(isempty(x0_min) || ~isnumeric(x0_min) || (x0_min == -Inf))
           x0_min = input([title ': ']);

           if(isempty(x0_min) || ~isnumeric(x0_min) || (x0_min == -Inf))
               fprintf(2, [title ' should be a number.\n']);
           end
        end
        x0_max = 136.585;%Inf;
        title = 'Maximum value for possible positions of amplitudes';
        while(isempty(x0_max) || ~isnumeric(x0_max) ...
                || (x0_max < x0_min) || (x0_max == Inf))
           x0_max = input([title ': ']);

           if(isempty(x0_max) || ~isnumeric(x0_max) ...
                || (x0_max < x0_min) || (x0_max == Inf))
               fprintf(2, [title ' should be a number.\n']);
           end
        end
    end

    % Calculate P(N|D,I) for N = n_min to N = n_max to find max probability
    for n = n_min:n_max
        % no calibrated value for eta
        if(etac == -1)
%           cftool(dataVEval(1,:), dataVEval(2,:))
%           pause;
            if(n==3)
                eta = 0.5;
            else
                eta = 0.58;
            end
            %eta = -1;
            while(isempty(eta) || ~isnumeric(eta) || (eta < 0) || (eta > 1))
               eta = input('By fitting tool calculated value for eta: ');

               if(isempty(eta) || ~isnumeric(eta) || (eta < 0) || (eta > 1))
                   fprintf(2, 'Eta should be a number between 0 and 1.\n');
               end
            end
        else
            eta = etac;
        end

        % (n-nX0) positions of amplitudes are to be estimated
        if(nX0 < n)
            x0_step = mstep;%-1;
            title = 'Step width/ precision for position of amplitude estimation'
            while(isempty(x0_step) || ~isnumeric(x0_step) ...
```

```
            || (x0_step < mstep) || (x0_step <= 0))
        x0_step = input([title '(measuring precision: ' ms '): ']);

        if(isempty(x0_step) || ~isnumeric(x0_step) ...
            || (x0_step < mstep) || (x0_step <= 0))
         fprintf(2, [title 'should be a positive number >= ' ms '.\n']);
        end
    end
    pxLimits = [x0_min x0_step x0_max];

    tic
    LF = n_pApw(eta, dataVEval(1,:), dataVEval(2,:), n, x0, ...
                pALimits, pwLimits, pxLimits);
    toc
elseif(nX0 == n)
    tic
    LF = n_pApw(eta, dataVEval(1,:), dataVEval(2,:), n, x0, ...
                pALimits, pwLimits);
    toc
end

%Marginalisation

if(nX0 == n) % parameter: A, w, LF:dim(1..n) = A, dim((n+1)..n*2) = w
    % estimation of A1
    projLF = squeeze(sum(LF, n*2));
    for m = n*2-1:-1:2
        projLF = squeeze(sum(projLF,m));
    end
    figure('Name','Maximum likelihood for A1', 'NumberTitle', 'off');
    plot(a_min:a_step:a_max, projLF, '-o');
    xlabel('estimation of amplitude for 1st peak');
    ylabel('likelihood function value');

    % estimation of A2
    projLF = squeeze(sum(LF, 1));
    for m = n*2-1:-1:2
        projLF = squeeze(sum(projLF,m));
    end
    figure('Name','Maximum likelihood for A2', 'NumberTitle', 'off');
    plot(a_min:a_step:a_max, projLF, '-o');
    xlabel('estimation of amplitude for 2nd peak');
    ylabel('likelihood function value');

    % estimation of A3
    projLF = squeeze(sum(LF, 1));
    for m = n*2-1:-1:3
        projLF = squeeze(sum(projLF,m));
    end
    projLF = squeeze(sum(projLF,1));
    figure('Name','Maximum likelihood for A3', 'NumberTitle', 'off');
    plot(a_min:a_step:a_max, projLF, '-o');
```

```
        xlabel('estimation of amplitude for 3rd peak');
        ylabel('likelihood function value');


        % estimation of w1
        projLF = squeeze(sum(LF, 1));
        for m = n*2-1:-1:4
            projLF = squeeze(sum(projLF,1));
        end
        projLF = squeeze(sum(projLF,2));
        projLF = squeeze(sum(projLF,2));
        figure('Name','Maximum likelihood for FWHM 1',...
                    'NumberTitle', 'off');
        plot(w_min:w_step:w_max, projLF, '-o');
        xlabel('estimation of FWHM for 1st peak');
        ylabel('likelihood function value');

        % estimation of w2
        projLF = squeeze(sum(LF, 1));
        for m = n*2-1:-1:3
            projLF = squeeze(sum(projLF,1));
        end
        projLF = squeeze(sum(projLF,2));
        figure('Name','Maximum likelihood for FWHM 2',...
                    'NumberTitle', 'off');
        plot(w_min:w_step:w_max, projLF, '-o');
        xlabel('estimation of FWHM for 2nd peak');
        ylabel('likelihood function value');

        % estimation of w3
        projLF = squeeze(sum(LF, 1));
        for m = n*2-1:-1:2
            projLF = squeeze(sum(projLF,1));
        end
        figure('Name','Maximum likelihood for FWHM 3',...
                    'NumberTitle', 'off');
        plot(w_min:w_step:w_max, projLF, '-o');
        xlabel('estimation of FWHM for 3rd peak');
        ylabel('likelihood function value');

        %P(N|D,I)
        tgral = squeeze(sum(LF, n*2));
        for m = n*2-1:-1:1
            tgral = squeeze(sum(tgral,m));
        end
        div = ((a_max - a_min)*(w_max - w_min))^double(n);

    % not all positions of amplitudes are given,
    % (n-nX0) positions of amplitudes are to be estimated
    elseif(nX0 < n) % parameter: A, w, x0
        %LF matrix :
        %dim(1..n) = A,
```

```
            %dim((n+1)..n*2) = w,
            %dim((n*2+1)..(n*3-nX0)) = x0

            %P(N|D,I)
            tgral = squeeze(sum(LF, n*2+(n-nX0)));
            for m = n*2+(n-nX0)-1:-1:1
                tgral = squeeze(sum(tgral,m));
            end
            div = ((a_max - a_min)*(w_max - w_min))^double(n)...
                                        *(x0_max-x0_min)^double(n-nX0);
        end

        postN = (double(factorial(n))/ div) * tgral;
        fprintf('%s %3.3f, %d %s -> P(%d | D, I) = %10.7g\n',...
                    'eta =', eta, nX0, 'positions of amplitudes given', ...
                    n, postN);
    end %for(calculate P(N| D,I) for N = n_min,..., n_max)

    ok = input('Do you want to evaluate data within another interval?(y/n): ',...
                's');

    evalFin = strcmp(ok, 'n');
end %while(end evaluations)

end



function lFunc = n_pApw(eta, x, f, n, x0, pALimits, pwLimits, pxLimits)
%
% lFunc - matrix of likelihood function values for sample points
%
% eta - proportion of lorentz function within pseudo voigt profile [0;1]
%   x - x values of measured data
%   f - y values of measured data
%   n - number of peaks to do estimation for
%  x0 - previously estimated values for positions of amplitudes
% pALimits - min value, step width, max value of amplitudes
% pwLimits - min value, step width, max value of FWHMs
% pxLimits - min value, step width, max value of positions of amplitudes
%            for estimation in case number of given x0 < n
%
if(n == 3 && numel(x0) == 3)
    if(eta >= 0 && eta <= 1) % given value for eta
        chi2 = n3_pApw(eta, x, f, x0, pALimits, pwLimits);
    else
        chi2 = n3Eta_pApw(x, f, x0, pALimits, pwLimits);
    end
elseif(n == 4 && numel(x0) == 3)
    if(eta >= 0 && eta <= 1) % given value for eta
        chi2 = n4_pApw(eta, x, f, x0, pALimits, pwLimits, pxLimits);
    else
```

```
        chi2 = n4Eta_pApw(x, f, x0, pALimits, pwLimits, pxLimits);
    end
end
%calculate likelihood-matrix
lFunc = exp(-0.5 * chi2);
end

function chi2 = n3_pApw(eta, x, f, x0, pALimits, pwLimits)
%   chi2 - matrix with chi^2 values
%        (square deviations between sample points and measuring points)
%
% eta - proportion of lorentz function within pseudo voigt profile [0;1]
%   x - x values of measured data
%   f - y values of measured data
%  x0 - previously estimated values for positions of amplitudes
% pALimits - min value, step width, max value of amplitudes
% pwLimits - min value, step width, max value of FWHMs
%

a_min = pALimits(1);
a_step = pALimits(2);
a_max = pALimits(3);

w_min = pwLimits(1);
w_step = pwLimits(2);
w_max = pwLimits(3);

%initial indices for matrix
i = 1; j = 1; k = 1;
l = 1; m = 1; n = 1;

%initialize minChi-matrix
minChi2 = zeros(1,7);
minChi2(7) = Inf;

%generate chi2-matrix
for pA1 = a_min:a_step:a_max
    for pA2 = a_min:a_step:a_max
        for pA3 = a_min:a_step:a_max
            for pw1 = w_min:w_step:w_max
                for pw2 = w_min:w_step:w_max
                    for pw3 = w_min:w_step:w_max
                        fAxw = n3_pvpFunc(x, eta, pA1, pA2, pA3,...
                               x0(1), x0(2), x0(3), pw1, pw2, pw3);
                        chi2(i,j,k,l,m,n) = (f - fAxw)*(f - fAxw)';
                        if (chi2(i,j,k,l,m,n) < minChi2(7))
                            minChi2(1) = pA1; minChi2(2) = pA2;
                            minChi2(3) = pA3;
                            minChi2(4) = pw1; minChi2(5) = pw2;
                            minChi2(6) = pw3;
                            minChi2(7) = chi2(i,j,k,l,m,n);
                        end %if(chi2 < minChi2)
```

```
                    n = n + 1;
                end %for(pw3)
                m = m + 1; n = 1;
            end %for(pw2)
            l = l + 1; m = 1;
        end %for(pw1)
        k = k + 1; l = 1;
    end %for(pA3)
    j = j + 1; k = 1;
end %for(pA2)
i = i + 1; j = 1;
end %for(pA1)
a_values = ['A1=␣' num2str(minChi2(1)) ',␣A2=␣' num2str(minChi2(2))];
a_values = [a_values ',␣A3=␣' num2str(minChi2(3))];
w_values = [',␣w1=␣' num2str(minChi2(4)) ',␣w2=␣' num2str(minChi2(5))];
w_values = [w_values ',␣w3=␣' num2str(minChi2(6))];
fprintf(['Chi2_min:␣' a_values w_values '\n']);
end

function chi2 = n3Eta_pApw(x, f, x0, pALimits, pwLimits)
%   chi2 - matrix with chi^2 values
%       (square deviations between sample points and measuring points)
%
%   x - x values of measured data
%   f - y values of measured data
%  x0 - previously estimated values for positions of amplitudes
% pALimits - min value, step width, max value of amplitudes
% pwLimits - min value, step width, max value of FWHMs
%

a_min = pALimits(1);
a_step = pALimits(2);
a_max = pALimits(3);

w_min = pwLimits(1);
w_step = pwLimits(2);
w_max = pwLimits(3);

%initial indices for matrix
i = 1;
j = 1; k = 1; l = 1;
m = 1; n = 1; o = 1;

%initialize minChi-matrix
minChi2 = zeros(1,8);
minChi2(8) = Inf;

%generate chi2-matrix
for eta = 0:0.01:1
    for pA1 = a_min:a_step:a_max
        for pA2 = a_min:a_step:a_max
            for pA3 = a_min:a_step:a_max
```

```
                for pw1 = w_min:w_step:w_max
                    for pw2 = w_min:w_step:w_max
                        for pw3 = w_min:w_step:w_max
                            fAxw = n3_pvpFunc(x, eta, pA1, pA2, pA3,...
                                    x0(1), x0(2), x0(3), pw1, pw2, pw3);
                            chi2(i,j,k,l,m,n,o) = (f - fAxw)*(f - fAxw)';
                            if (chi2(i,j,k,l,m,n,o) < minChi2(8))
                                minChi2(1) = eta;
                                minChi2(2) = pA1; minChi2(3) = pA2;
                                minChi2(4) = pA3;
                                minChi2(5) = pw1; minChi2(6) = pw2;
                                minChi2(7) = pw3;
                                minChi2(8) = chi2(i,j,k,l,m,n,o);
                            end %if(chi2 < minChi2)
                            o = o + 1;
                        end %for(pw3)
                        n = n + 1; o = 1;
                    end %for(pw2)
                    m = m + 1; n = 1;
                end %for(pw1)
                l = l + 1; m = 1;
            end %for(pA3)
            k = k + 1; l = 1;
        end %for(pA2)
        j = j + 1; k = 1;
    end %for(pA1)
    i = i + 1; j = 1;
end %for(eta)
a_values = ['A1=␣' num2str(minChi2(2)) ',␣A2=␣' num2str(minChi2(3))];
a_values = [a_values ',␣A3=␣' num2str(minChi2(4))];
w_values = [',␣w1=␣' num2str(minChi2(5)) ',␣w2=␣' num2str(minChi2(6))];
w_values = [w_values ',␣w3=␣' num2str(minChi2(7))];
fprintf(['Chi2_min:␣eta=␣' num2str(minChi2(1)) a_values w_values '\n']);
end

function chi2 = n4_pApw(eta, x, f, x0, pALimits, pwLimits, pxLimits)
%   chi2 - matrix with chi^2 values
%       (square deviations between sample points and measuring points)
%
% eta - proportion of lorentz function within pseudo voigt profile [0;1]
%   x - x values of measured data
%   f - y values of measured data
%  x0 - previously estimated values for positions of amplitudes
% pALimits - min value, step width, max value of amplitudes
% pwLimits - min value, step width, max value of FWHMs
% pxLimits - min value, step width, max value of position of amplitude
%
a_min = pALimits(1);
a_step = pALimits(2);
a_max = pALimits(3);

w_min = pwLimits(1);
```

```
w_step = pwLimits(2);
w_max = pwLimits(3);

x0_min = pxLimits(1);
x0_step = pxLimits(2);
x0_max = pxLimits(3);

%initial indices for matrix
i = 1; j = 1; k = 1; l = 1;
m = 1; n = 1; o = 1; p = 1; q = 1;

%initialize minChi-matrix
minChi2 = zeros(1,10);
minChi2(10) = Inf;

%generate chi2-matrix
for pA1 = a_min:a_step:a_max
    for pA2 = a_min:a_step:a_max
        for pA3 = a_min:a_step:a_max
            for pA4 = a_min:a_step:a_max
                for pw1 = w_min:w_step:w_max
                    for pw2 = w_min:w_step:w_max
                        for pw3 = w_min:w_step:w_max
                            for pw4 = w_min:w_step:w_max
                                for px4 = x0_min:x0_step:x0_max
                                fAxw = n4_pvpFunc(x, eta, pA1, pA2, pA3, ...
                                        pA4, x0(1), x0(2), x0(3), px4, pw1,...
                                        pw2, pw3, pw4);
                          chi2(i,j,k,l,m,n,o,p,q) = (f - fAxw)*(f - fAxw)';
                          if (chi2(i,j,k,l,m,n,o,p,q) < minChi2(10))
                              minChi2(1) = pA1; minChi2(2) = pA2;
                              minChi2(3) = pA3; minChi2(4) = pA4;
                              minChi2(5) = pw1; minChi2(6) = pw2;
                              minChi2(7) = pw3; minChi2(8) = pw4;
                              minChi2(9) = px4;
                              minChi2(10) = chi2(i,j,k,l,m,n,o,p,q);
                          end %if(chi2 < minChi2)
                                    q = q + 1;
                                end %for(px4)
                                p = p + 1; q = 1;
                            end %for(pw4)
                            o = o + 1; p = 1;
                        end %for(pw3)
                        n = n + 1; o = 1;
                    end %for(pw2)
                    m = m + 1; n = 1;
                end %for(pw1)
                l = l + 1; m = 1;
            end %for(pA4)
            k = k + 1; l = 1;
        end %for(pA3)
        j = j + 1; k = 1;
```

```
    end %for(pA2)
    i = i + 1; j = 1;
end %for(pA1)
a_values = ['A1=␣' num2str(minChi2(1)) ',␣A2=␣' num2str(minChi2(2))];
a_values = [a_values ',␣A3=␣' num2str(minChi2(3)) ',␣A4=␣' num2str(minChi2(4))];
w_values = [',␣w1=␣' num2str(minChi2(5)) ',␣w2=␣' num2str(minChi2(6))];
w_values = [w_values ',␣w3=␣' num2str(minChi2(7)) ',␣w4=␣' num2str(minChi2(8))];
x4_value = [',␣x4=␣' num2str(minChi2(9))];
fprintf(['Chi2_min:␣' a_values w_values x4_value '\n']);
end

function chi2 = n4Eta_pApw(x, f, x0, pALimits, pwLimits, pxLimits)
%   chi2 - matrix with chi^2 values
%       (square deviations between sample points and measuring points)
%
%   x - x values of measured data
%   f - y values of measured data
%  x0 - previously estimated values for positions of amplitudes
% pALimits - min value, step width, max value of amplitudes
% pwLimits - min value, step width, max value of FWHMs
% pxLimits - min value, step width, max value of position of amplitude
%
a_min = pALimits(1);
a_step = pALimits(2);
a_max = pALimits(3);

w_min = pwLimits(1);
w_step = pwLimits(2);
w_max = pwLimits(3);

xa_min = pxLimits(1);
xa_step = pxLimits(2);
xa_max = pxLimits(3);

%initial indices for matrix
i = 1; j = 1; k = 1; l = 1; m = 1; n = 1; o = 1; p = 1; q = 1; r = 1;

%initialize minChi-matrix
minChi2 = zeros(1,11); minChi2(11) = Inf;

%generate chi2-matrix
for eta = 0:0.01:1
    for pA1 = a_min:a_step:a_max
        for pA2 = a_min:a_step:a_max
            for pA3 = a_min:a_step:a_max
                for pA4 = a_min:a_step:a_max
                    for pw1 = w_min:w_step:w_max
                        for pw2 = w_min:w_step:w_max
                            for pw3 = w_min:w_step:w_max
                                for pw4 = w_min:w_step:w_max
                                    for px4 = xa_min:xa_step:xa_max
                        fAxw = n4_pvpFunc(x, eta, pA1, pA2, pA3, pA4, ...
```

```
                                x0(1), x0(2), x0(3), px4, pw1, pw2, pw3, pw4);
                            chi2(i,j,k,l,m,n,o,p,q,r) = (f - fAxw)*(f - fAxw)';
                            if (chi2(i,j,k,l,m,n,o,p,q,r) < minChi2(11))
                                minChi2(1) = eta;
                                minChi2(2) = pA1; minChi2(3) = pA2;
                                minChi2(4) = pA3; minChi2(5) = pA4;
                                minChi2(6) = pw1; minChi2(7) = pw2;
                                minChi2(8) = pw3; minChi2(9) = pw4;
                                minChi2(10) = px4;
                                minChi2(11) = chi2(i,j,k,l,m,n,o,p,q,r);
                            end %if(chi2 < minChi2)
                                        r = r + 1;
                                    end %for(px4)
                                    q = q + 1; r = 1;
                                end %for(pw4)
                                p = p + 1; q = 1;
                            end %for(pw3)
                            o = o + 1; p = 1;
                        end %for(pw2)
                        n = n + 1; o = 1;
                    end %for(pw1)
                    m = m + 1; n = 1;
                end %for(pA4)
                l = l + 1; m = 1;
            end %for(pA3)
            k = k + 1; l = 1;
        end %for(pA2)
        j = j + 1; k = 1;
    end %for(pA1)
    i = i + 1; j = 1;
end %for(eta)
a_values = ['A1=' num2str(minChi2(2)) ', A2=' num2str(minChi2(3))];
a_values = [a_values ', A3=' num2str(minChi2(4)) ', A4=' num2str(minChi2(5))];
w_values = [', w1=' num2str(minChi2(6)) ', w2=' num2str(minChi2(7))];
w_values = [w_values ', w3=' num2str(minChi2(8)) ', w4=' num2str(minChi2(9))];
x4_value = [', x4=' num2str(minChi2(10))];
fprintf(['Chi2_min: eta=' num2str(minChi2(1)) a_values w_values x4_value '\n'])
end
```

Ich habe die Funktionen n1_pvpFunc bis n7_pvpFunc in gleicher Weise erstellt, wie die hier abgebildeten zwei Funktionen für ein Spektrum aus drei und vier Peaks.

```
function y = n3_pvpFunc(x, eta, pA1, pA2, pA3, px1, px2, px3, pw1, pw2, pw3)

y = pA1.*(eta./(1 + ((x - px1)./(pw1/2)).^2) ...
        + (1 - eta).*exp(-log(2).*((x - px1)./(pw1/2)).^2))...
    + pA2.*(eta./(1 + ((x - px2)./(pw2/2)).^2) ...
        + (1 - eta).*exp(-log(2).*((x - px2)./(pw2/2)).^2))...
    + pA3.*(eta./(1 + ((x - px3)./(pw3/2)).^2) ...
        + (1 - eta).*exp(-log(2).*((x - px3)./(pw3/2)).^2));
end


function y = n4_pvpFunc(x, eta, pA1, pA2, pA3, pA4, px1, px2, px3, px4,...
    pw1, pw2, pw3, pw4)

y = pA1.*(eta./(1 + ((x - px1)./(pw1/2)).^2)...
        + (1 - eta).*exp(-log(2).*((x - px1)./(pw1/2)).^2)) ...
    + pA2.*(eta./(1 + ((x - px2)./(pw2/2)).^2)...
        + (1 - eta).*exp(-log(2).*((x - px2)./(pw2/2)).^2)) ...
    + pA3.*(eta./(1 + ((x - px3)./(pw3/2)).^2)...
        + (1 - eta).*exp(-log(2).*((x - px3)./(pw3/2)).^2)) ...
    + pA4.*(eta./(1 + ((x - px4)./(pw4/2)).^2)...
        + (1 - eta).*exp(-log(2).*((x - px4)./(pw4/2)).^2));
end
```

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe;
die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.
Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und ist auch noch nicht veröffentlicht.

Christiane Schöpflin
Heidelberg, November 2016